An Efficient Ensemble Algorithm for Boosting *k*-Nearest Neighbors Classification Performance via Feature Bagging

Huu-Hoa Nguyen
College of Information and Communication Technology, Can Tho University, Vietnam

Abstract*—This paper proposes a novel ensemble algorithm aimed at improving the performance of *k*-Nearest Neighbors (KNN) classification by incorporating feature bagging techniques, which help overcome the inherent limitations of KNN in Big Data scenarios. The proposed algorithm, termed FBE (Feature Bagging-based Ensemble), employs an efficient ensemble strategy with sorted feature subset techniques to reduce the time complexity from linear to logarithmic. By focusing on essential features during iterative training and utilizing a binary search in the testing phase, FBE boosts computational efficiency and accuracy in high-dimensional and imbalanced datasets. Our study rigorously evaluates the proposed FBE algorithm against traditional KNN, Random Forest (RF), and AdaBoost algorithms across ten benchmark datasets from the UCI Machine Learning Repository. The experimental results demonstrate that FBE not only outperforms the conventional KNN and AdaBoost across all evaluated metrics (accuracy, precision, recall, and F1 score) but also shows competitive performance compared to RF. Specifically, FBE exhibits remarkable improvements in datasets characterized by high dimensionality and class imbalances. The main contributions of this research include the development of an adaptive KNN framework that addresses the typical computational demands and vulnerability to noise in the data, making it well-suited for large-scale datasets. The ensemble methodology within FBE also helps reduce overfitting, a common challenge in standard KNN models, by diversifying the decision-making process across multiple data subsets. This strategy ensures robustness and reliability, positioning FBE as a suitable tool for classification tasks in diverse domains such as healthcare and image processing.

***Keywords*—Bagging; ensemble; feature; k-nearest neighbors**

I. Introduction

Machine learning (ML) significantly improves our ability to analyze large data sets and extract actionable insights in various sectors, including healthcare and financial services. However, integrating ML with Big Data presents complex challenges, such as managing the vast volumes and varieties of data that could exceed the capabilities of traditional processing methods. These challenges can complicate the training and fine-tuning of ML models, impacting their scalability. Furthermore, Big Data can exacerbate issues like overfitting, where models perform well on training data but poorly on new data. To overcome these difficulties, there is a pressing need for advanced algorithms tailored for large-scale data, as well as techniques for effective dimensionality reduction and rigorous model validation.

The spectrum of machine learning models is varied, each designed to meet specific data characteristics and analytical requirements. Probabilistic models, like Bayesian networks, excel at managing data uncertainty and variability but require significant computational resources [1]. Regression models are essential for predicting continuous variables and provide clear interpretations, though they may oversimplify complex relationships [2]. Architectural models, such as neural networks, excel in pattern recognition and addressing non-linear challenges, but they require substantial data and computational resources and often lack clarity in their decision-making processes [3]. Similarly, distance-based models like the *k*-Nearest Neighbors are effective in classification tasks that rely on proximity measures, yet struggle with high-dimensional data due to the curse of dimensionality [4]. Each model type offers unique advantages and limitations, necessitating careful selection to align with specific goals and constraints.

In this research, we focus on distance/similarity-based ML models, specifically the *k*-Nearest Neighbors (KNN) algorithm [4]. KNN classifies new instances based on the most frequent class among the closest neighbors within the feature space. This inherently non-parametric and lazy learning model memorizes the training data rather than constructing a definitive model, enabling high adaptability and immediate response to new data. Despite its simplicity and effectiveness, KNN faces several challenges. As a lazy learner that retains the entire dataset, KNN's computational demands increase with the size of the data, limiting its use in large-scale datasets. The algorithm's accuracy is also compromised by noisy or irrelevant features that can distort distance measurements, leading to inaccurate classifications. Moreover, choosing an optimal number of neighbors (k) is critical; too few can lead to overfitting, while too many may cause underfitting. Additionally, KNN struggles with datasets that exhibit significant class imbalances, potentially biasing predictions toward majority classes.

Our study explores enhancements to the traditional KNN approach to address scalability issues, thereby optimizing its efficiency without compromising accuracy, making it particularly suitable for Big Data applications. Specifically, this paper introduces a novel algorithm called FBE (Feature Bagging-based Ensemble), designed to boost the performance of KNN classification through feature bagging. This method significantly reduces the traditional model's time complexity from linear to logarithmic by sorting data subsets during the training phase and utilizing an efficient binary search in the testing phase, making it particularly suitable for Big Data applications. We rigorously evaluated the proposed FBE algorithm on ten benchmark datasets from the UCI Machine Learning Repository. The experimental results demonstrated significant improvements in classification performance over traditional KNN and AdaBoost, and were competitive with the Random Forest classifier. Our comprehensive experiments highlight FBE's potential for handling complex, imbalanced, or high-dimensional datasets. The algorithm experimentally excels across various metrics, including accuracy, precision, recall, and F1 score, underscoring its robustness and adaptability. Through detailed evaluation and comparison with standard machine learning models, FBE has proven its effectiveness and versatility, addressing a wide range of challenging datasets.

The remainder of this paper is structured as follows. Section II explores literature surveys and synthesis. Section III details the proposed algorithm, outlining its methodology and theoretical underpinnings, whereas Section IV is dedicated to experimental validation. Finally, Section V concludes the paper with a summary of our findings and future research.

II. Literature Surveys and Synthesis

In the field of machine learning, particularly with respect to *k*-Nearest Neighbors (KNN) algorithms, considerable progress has been made in addressing the computational challenges inherent to KNN. This section examines a variety of methods developed to enhance KNN's performance and efficiency.

Among various strategies to enhance KNN, dimensionality reduction is particularly impactful. It plays a crucial role in improving the efficiency of KNN by transforming high-dimensional data into a more manageable format without significant information loss. One approach [5] employs an Extreme Learning Machine (ELM) to simplify complex data into a more accessible feature space. ELM, a supervised machine learning method with a single hidden layer, is noted for its rapid processing capabilities. However, it is also sensitive to noise and heavily depends on the random selection of weights and biases, which can limit its effectiveness. Another method [6] uses Mutual Information (MI) to enhance the efficiency of dimensionality reduction and employs General Purpose Graphics Processing Units to parallelize the nearest neighbor search process. While effective, this method requires additional hardware resources, which may not be practical in all settings.

To mitigate the resource consumption challenges associated with kNN, numerous researchers have explored tree-based solutions as a common strategy. These methods typically involve selecting a splitting criterion to construct a tree, often a binary tree, which organizes the dataset in a way that accelerates the search for the nearest neighbor. Several innovative tree-based models have been developed, such as the Combi Tree, which offers adaptive approaches to optimizing KNN.

The Combi Tree, developed from a binary search tree [7], segments the data points into clusters and uses a hash table to compress each cluster. These clusters are then combined to form the Combi Tree. However, this approach operates exclusively within Hamming space, a high-dimensional space suited for binary data, making it less effective for other data types or similarity measures outside this space. Another strategy constructs a Binary Search Tree (BST) based on the norms of data points [8]. This involves a partitioning scheme that uses norms to distribute data points evenly within the BST, although this method may struggle with skewness in data distributions.

Furthermore, a novel BST method [9] incorporates a scaling factor to improve search speed, particularly beneficial for managing large datasets where traditional binary search trees may be cumbersome and inefficient. This method adjusts the size of search intervals using the scaling factor as the search progresses, allowing for a rapid narrowing of the search space and achieving logarithmic time complexity. However, the validation of this method has been limited to synthetic data, with its effectiveness in real-world scenarios yet to be confirmed.

While tree-based methods focus on structural optimizations, another approach involves refining the data itself through clustering. One method [10] utilizes the k-means clustering algorithm to segment the dataset, removing data points that have minimal impact on accuracy. During the testing phase, this method determines the cluster to which a given instance belongs and performs KNN within that specific subset. However, this pruning technique may not achieve optimal results in scenarios where the decision boundary is non-linear or the dataset is inherently noisy.

To further refine this strategy, another study [11] uses clustering to reduce the number of data points required for each query. This method introduces a technique of region division to further limit the search space. It divides the space into several smaller regions and considers only the data points within the region containing the query point for the KNN search. Nevertheless, the clustering can become computationally demanding with high-dimensional data, restricting the scalability of these methods as the number of features increases.

Building on the idea of clustering, the KNN Tree method [12] combines tree-based and clustering strategies to further enhance efficiency. It constructs a decision tree (DT) up to a specified depth and then applies KNN to the remaining subset of the dataset. This hybrid algorithm effectively reduces the number of samples required for KNN, thereby enhancing the model's efficiency. Notably, this method surpasses the performance of both standalone DT and traditional KNN, showing significant improvements in managing large-scale datasets.

To contextualize these advancements, Table I compares these various approaches, underscoring the trade-offs and enhancements that our algorithm introduces. These comparisons elucidate the trade-offs involving speed, scalability, noise sensitivity, and data specificity among the different methods. Although each method significantly enhances the efficiency of KNN, they also introduce specific limitations that may affect their utility depending on application scenarios.

TABLE. I. COMPARATIVE ANALYSIS ACROSS VARIOUS RELATED METHODS

Methods	Advantage	Drawback
ELM based [5]	Increases processing speed by simplifying data representation.	Sensitive to noise, affecting the robustness of classifications.
Pknn-mifs [6]	Speeds up the KNN process through parallel processing.	Requires additional hardware resources, increasing costs.
Combi tree [7]	Achieves logarithmic complexity, suitable for binary data.	Restricted to applications within Hamming space only.
Norm based [8]	Effective in environments with uniform data distribution.	Limited effectiveness in non-uniform or skewed data sets.
BST based [9]	Achieves logarithmic complexity, optimizing search times.	Performance primarily validated on synthetic, not real-world, data.
EDP [10]	Enhances speed by efficiently pruning unnecessary data.	Performance declines with non-linear data scenarios.
SRBC [11]	Provides faster execution compared to traditional KNN.	Does not scale well with high-dimensional data.
KNNTree [12]	Achieves logarithmic time complexity, enhancing efficiency.	Performance heavily dependent on correct hyper-parameter tuning.

III. PROPOSED ALGORITHM (FBE)

A. *General Idea of FBE*

The general idea behind the proposed FBE algorithm is to transform the computationally expensive KNN into a more efficient and scalable model by employing the power of approximation-based sorting and ensemble learning. Below, we explore the foundational concepts that underpin the FBE.

1) Dimensional selection and data sorting: FBE starts with the strategic selection and use of data dimensions. For example, consider a dataset represented in a three-dimensional space, as illustrated in Fig. 1. Within the FBE framework, dimensions are selected randomly, such as dimensions D1 and D3 for a specific iteration. The selection of dimensions is critical as it influences the subsequent sorting of the dataset. If D3 shows a stronger correlation with the class labels than D1, it becomes the primary axis for sorting. This sorting process, illustrated in Fig. 2, is essential as it reorganizes the dataset to align more closely with the inherent data structure, thus enabling more efficient searches during the testing phase.

2) Approximation-based search and ensemble techniques: FBE uses approximation-based search techniques to reduce the time complexity traditionally associated with KNN, typically where *n* is the number of instances and *d* is the dimensionality of the data. By sorting the data according to selected dimensions that show a consistent relationship with the target variable, the algorithm paves the way for a binary search. This search method significantly reduces the search space from linear to logarithmic time complexity, making it feasible to handle large datasets effectively.

The integration of ensemble techniques further enhances the FBE algorithm. By repeating the selection and sorting process multiple times, each with potentially different dimensions, the algorithm creates a diverse set of data views, each organized according to the most informative feature of that iteration. This ensemble of sorted subsets not only reduces the risk of bias in the model but also improves overall accuracy through collective decision-making during the testing phase.

3) Synergistic benefits: The synergy between sorted subset selection and ensemble strategies results in a robust algorithm that not only increases computational efficiency but also sustains, if not improves, classification effectiveness. The ensemble approach reduces variance and potential overfitting by integrating multiple independent evaluations of nearest neighbors, each from slightly different perspectives of the data. As a result, FBE presents a compelling alternative to conventional KNN, especially in scenarios involving large-scale datasets with complex, non-linear relationships among features.

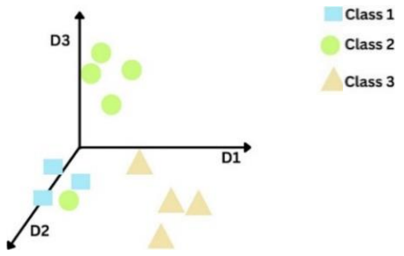


Fig. 1. Data points in three dimensions

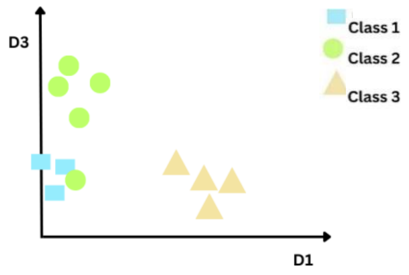


Fig. 2. Data points in randomly selected two dimensions

B. *FBE in Training Phase*

The training phase of FBE is methodically outlined in Algorithm 1 and consists of three primary steps, as follows.

1) Step 1: Initialization: The algorithm begins by initializing an empty set S_F which will eventually store subsets of the training data alongside their corresponding sorting features. This set plays a pivotal role in the ensemble strategy, facilitating a diverse array of simplified datasets for efficient neighbor searches during the testing phase.

2) Step 2: Iterative processing: The core of Algorithm 1 operates over *m* iterations, reflecting the ensemble nature of FBE. Each iteration is designed to create a unique subset of the data, focusing on different features to capture various characteristics of the data:

- Feature Selection: In each iteration, a subset of features, X', is randomly selected from the feature set *X*. This randomness introduces diversity in the features considered across different iterations, which is fundamental to the ensemble approach.
- Best Feature Determination: For each selected feature a, its mutual information with the target labels y is calculated. Mutual information, denoted as MI(a,y), measures the amount of information one variable contains about another, thus helping to identify the most predictive features. Mathematically, MI is defined as:

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right),$$

where *p*(*x*,*y*) is the joint probability distribution of *X* and *Y*, and *p*(*x*) and *p*(*y*) are the marginal distributions of *X* and *Y*, respectively.

- The feature *a* with the highest *MI* value is selected as the best feature *b*. This feature is considered the most effective at classifying data points for that iteration, guiding the sorting process.

3) Step 3: Sorting and storing

- Once the best feature *b* is identified, the subset of the data X_t, corresponding to X', along with the labels y_t, are sorted based on *b*. This sorting is pivotal as it rearranges the data points so that similar values (and thus potentially similar classes) are positioned closer together, drastically enhancing the efficiency of neighbor searches in high-dimensional spaces.

The sorted subset along with its metadata (features used and the best feature) is encapsulated into a tuple $val = (X_t, y_t, X', b)$ and added S_F. Each tuple in S_F represents a different "view" or "model" of the dataset, optimized for quick searching within the framework of the proposed ensemble method.

By the end of *m* iterations, S_F contains multiple sorted versions of subsets of the training set, each optimized differently based on the selected features. This setup allows the FBE algorithm during the testing phase to quickly locate the nearest neighbors by leveraging the pre-processed, efficiently sorted data structures, greatly reducing computational overhead and time complexity compared to traditional KNN approaches.

In essence, Algorithm 1 lays the foundational work for FBE, ensuring that the ensemble method not only maintains high classification performance but also addresses the scalability issues often associated with KNN, particularly in large datasets.

Algorithm 1: FBE in the training phase

```
Input:
  • $X = \{x_1, x_2, ..., x_n\}$ : Set of feature vectors
  • $y = \{y_1, y_2, ..., y_n\}$ : Corresponding labels
  • $k$: Number of nearest neighbors
  • $m$: Number of iterations for ensemble
  • $g$: Grace parameter
Output:
  • $S_F$ : Set of sorted features and associated metadata
Procedure:
1. Initialize the sorted feature set $S_F$ to an empty set.
2. For each iteration $i$ from 1 to $m$:
     • Select a subset of features $X'$ randomly from $X$.
     • Initialize the best feature $b$ to none and the maximum mutual information $Max\_MI$ to -1.
     • For each feature $a$ in $X'$:
         ▪ Compute the mutual information $MI$ between $a$ and $y$.
         ▪ If $MI$ is greater than $Max\_MI$:
             - Update $Max\_MI$ with $MI$.
             - Set the best feature $b$ to $a$.
     • Select the subset of $X$ corresponding to $X'$ as $X_t$ and set $y_t = y$.
     • Sort $X_t$ and $y_t$ based on the values of the best feature $b$.
     • Create a tuple $val = (X_t, y_t, X', b)$ and add it to $S_F$
3. Return $S_F$
```

C. FBE in Testing Phase

The testing phase of FBE is methodically described in Algorithm 2. This representation of Algorithm 2 aligns with Algorithm 1 by directly utilizing the sorted subsets generated during the training phase, ensuring that the ensemble method efficiently employs the preprocessed data to enhance prediction accuracy and computational efficiency. This testing phase comprises four primary tasks, as follows.

Algorithm 2: FBE in the testing phase

```
Input:
    • $X_i$ : A given testing data point.
Output:
    • $C_i$ : Predicted class label for the testing data point $X_i$ .
Procedure:
1.  Initialize an empty list of predictions P to store the predicted labels from each iteration.
2.  For each iteration i from 1 to m (as set in the training phase):
      • Set initial search bounds low = 1 and high = n, where n is the total number of data points in the data subset.
      • Extract the sorted subset of features $X_t$ and the corresponding labels $y_i$ from the sorted feature set $S_F$ prepared in the training phase:
          ▪ $X_t = S_F[i]["features"]$
          ▪ $y_t = S_F[i]["labels"]$
          ▪ $X' = S_F[i][b]$
      • Determine the index in $X_t$ that best matches $X_i$ based on $X'$ , using binary search:
          ▪ While $low < high$:
              -  Compute $mid = low + (high - low)/2$.
              -  If $X_t[X'][mid] < X_i[X']$ then set $low = mid + 1$,
                 else set $high = mid - 1$.
      • After locating the nearest region, define the search interval within the sorted data:
          ▪ $left = \max(0, low - k - g)$
          ▪ $right = \min(n, low + k + g)$
      • Use traditional KNN to predict the class label from the subset $X_t[left : right]$ and
        $y_t[left : right]$ , and add the result to predictions P.
3.  After all iterations, determine the majority class label from P and return it as $C_i$ .
```

Legend:

- *k*: Number of nearest neighbors (defined in training phase).
- *g*: Grace parameter (defined in training phase).
- S_F : Sorted feature set, containing tuples of sorted feature subsets and their corresponding labels from the training phase.
- *m*: Number of ensemble iterations, aligning with the number of sorted subsets in S_F .

1) Initialization and prediction collection

- The algorithm begins by initializing an empty list of predictions, *P*, designed to collect the outcomes from each iteration. This facilitates an ensemble approach where multiple predictions are aggregated to determine the most likely class for a given testing instance, X_i .
- The ensemble method employed here ensures robustness in the predictions by averaging out biases that may be present in any single sorted subset of the training data.

2) Iterative binary search on sorted subsets

- For each iteration within the predefined number of ensemble iterations *m*, the algorithm processes through subsets of data that were sorted and stored during the training phase. These subsets are indexed from the structured set S_F, which contains key information such as the subset of features, the corresponding labels, and the feature that demonstrated the highest mutual information with the outcome during the training phase.

- The core of the testing phase is a binary search on the selected subset using the best feature *b* identified during training. This guides the search to efficiently locate the segment of the dataset where the testing instance might belong, based on feature similarity.
- The binary search algorithm adjusts the *low* and *high* pointers based on comparisons between $X_i[X']$ and the mid-point value of $X_t[X']$. This method drastically reduces the number of comparisons needed to locate the nearest region in the dataset from which the neighbors are selected.

3) Local nearest neighbor determination

- Once the approximate location of X_i is pinpointed in the sorted array, a local neighborhood is defined around this point. The size of this neighborhood is adjusted by the parameters *k* and *g*, where *k* denotes the number of nearest neighbors typically considered in KNN, and *g* allows for an expanded search buffer to mitigate the risk of missing potential nearest neighbors due to boundary effects or sparse regions within the dataset.
- Traditional KNN is then applied within this localized segment of the dataset to predict the class based on the majority vote among the *k*-nearest neighbors found in this region. This ensures the fundamental KNN principle of classifying based on the nearest data points is preserved, even within the ensemble method.

4) Aggregation and final prediction

- After cycling through all iterations, the predictions from each subset are aggregated to determine the final class label for X_i. The aggregation method typically involves selecting the majority class from the list of predictions, using the ensemble's diversity to provide a more accurate and stable prediction.
- This majority voting system across different predictive models reduces variance and improves the reliability of the classification, particularly in cases where individual models might have biases or perform poorly under specific data conditions.

In essence, Algorithm 2 enhances the traditional KNN approach by integrating an efficient binary search within strategically preprocessed subsets and utilizing an ensemble methodology to derive robust and accurate predictions. This approach not only speeds up the classification process significantly by reducing the number of distance calculations typically required in KNN but also leverages the diversity of multiple models to improve overall prediction accuracy. The combination of these strategies makes FBE particularly suitable for large-scale datasets where traditional KNN might struggle with scalability and performance.

D. Complexity Analysis of the FBE Algorithm

Understanding the computational complexity of FBE is essential for evaluating its efficiency, especially when compared to traditional KNN-based methods. This section examines the complexity of both the training and testing phases of the FBE, highlighting the improvements made by incorporating sorted subsets and ensemble techniques.

1) Complexity analysis of FBE in the training phase: The training phase of FBE involves selecting subsets of features, computing mutual information (*MI*) to identify the most informative features, and sorting these subsets for efficient search during testing. Let *n* be the number of instances, *d* the number of dimensions, *k* the number of nearest neighbors, *m* the number of iterations, and *g* the grace parameter.

- Feature selection and mutual information calculation:

During each iteration, a subset of features is selected randomly, which introduces variability but also necessitates a reassessment of the data structure for each subset. The mutual information calculation, which helps in selecting the best feature for sorting, typically has a complexity of $O(n)$ per feature. Considering that any or all features could be involved in the worst case, the complexity for this step is $O(n.d)$.

- Sorting:

After identifying the best feature, the subset is sorted based on this feature. Sorting a list of *n* elements generally consumes $O(n \log n)$ time. Since this is done for each dimensionally reduced subset (effectively each feature in the worst case), the complexity becomes $O(nd \log n)$.

- Overall complexity in the training phase:

Combining these factors, the complexity for each iteration is $O(nd + nd \log n)$, which simplifies to $O(nd \log n)$. Across *m* iterations, this results in a total complexity of $O(mnd \log n)$, representing a significant computational requirement but still more manageable than exhaustive pairwise comparisons across all features and instances.

2) Complexity analysis of FBE in the testing phase: The testing phase utilizes the pre-sorted subsets and a binary search mechanism to quickly locate potential nearest neighbors, significantly reducing the time required for each query.

- Binary search:

Binary search on a sorted subset has a complexity of $O(\log n)$, which is independent of the dimensionality *d* because the search is confined to the sorted dimension identified during the training phase.

- Local KNN computation:

Once the approximate location of the test instance is determined, a localized KNN search is performed within a segment defined by *k* and *g*. The computational cost for this localized search depends on the size of the segment but remains less than searching the entire dataset. The complexity for this step is approximated as $O(kd + dg + k^2 + kg)$, which simplifies to $O(k^2)$ in practical scenarios where *k* is much smaller than *n*.

- Overall complexity in the testing phase:

The combined complexity of the testing phase for *m* iterations is $O(m(\log n + k^2))$. This highlights a substantial efficiency over methods that require full dataset scans.

3) Space complexity analysis of FBE: The space complexity of FBE is primarily determined by the storage required for the sorted subsets and their associated metadata. For *m* iterations, storing each subset and its features requires $O(mnd)$ space, slightly higher than traditional KNN but justified by the significant gains in query time performance.

In summary, the FBE algorithm presents a well-optimized approach to KNN, with $O(mnd \log n)$ time complexity for training and $O(m(\log n + k^2))$ for testing. These improvements make FBE particularly suitable for large-scale datasets where the balance between accuracy, computational speed, and resource utilization is crucial. The algorithm effectively utilizes the strengths of ensemble methods and sorted data structures to enhance the scalability of nearest neighbor searches.

IV. Experimental Validation

A. Dataset Descriptions

To evaluate the effectiveness of FBE, a comprehensive selection of ten benchmark datasets was chosen, emphasizing the diversity and complexity inherent in real-world data. These datasets, predominantly sourced from the UCI Machine Learning Repository, are particularly suited to demonstrating the robust capabilities of FBE due to their varied challenges and characteristics.

Among the datasets selected, five are centered on medical applications, a domain where data complexity and the need for precision and reliability are crucial. These datasets include ECG, Diabetes, Lymphography, Fertility, and Breast Cancer, each presenting unique challenges due to their imbalance and the nature of the outcomes they seek to predict. For instance, the ECG dataset, with its extensive feature set, tests the algorithm's ability to handle large-scale data under circumstances where accuracy in predicting heart conditions can be lifesaving. On the other hand, datasets like Diabetes and Breast Cancer require the model to manage imbalanced data, where the prevalence of one class over another could bias the learning process, potentially leading to inaccurate diagnoses. Further complexity is introduced with the inclusion of datasets such as MNIST, which differ significantly in terms of dimensionality and class structure. The MNIST dataset, with its high-dimensional space composed of handwritten digit images, challenges the model to efficiently process and classify complex visual patterns. Additionally, the selection of datasets with smaller sample sizes mirrors typical scenarios where high-performance classification must be achieved despite limited data availability.

Details of the selected datasets, including specific features and class distributions, are meticulously catalogued in Table II. This table serves as a reference point for understanding the diverse data challenges that FBE is engineered to tackle, illustrating the algorithm's broad applicability and robust performance across a variety of complex scenarios.

B. Experimental Setup

1) Computational tools: The experiments were conducted on a Linux Fedora 32 operating system, using an Intel Core i7-4790 CPU at 3.6 GHz and equipped with 32 GB of RAM. This configuration, akin to a high-end personal computing setup, provides a stable and balanced environment suitable for both the development and evaluation phases.

TABLE. II. Datasets Used for Experiments

ID.	Dataset Name	Instance	Feature	Class
1	Breast Cancer	569	30	2
2	Lymphography	148	18	4
3	Fertility	100	8	2
4	ECG	109,446	187	5
5	Diabetes	768	8	2
6	Iris	150	4	3
7	Spambase	110,201	4	2
8	MNIST	70,000	784	10
9	Glass	214	9	6
10	Magic	19,020	10	2

Python was chosen as the primary programming language due to its extensive support for machine learning. Key Python libraries utilized in the setup include:

- Numpy: Facilitates efficient numerical computations with support for large, multi-dimensional arrays and matrices. This library is crucial for performance optimization in data-intensive applications.
- Pandas: Offers powerful data manipulation capabilities that simplify data cleaning, transformation, and analysis, essential for preparing datasets for machine learning.
- Scikit-Learn: Provides a wide array of machine learning algorithms and tools, making it indispensable for model training, evaluation, and comparison.

2) Model benchmarking and parameter setting: To contextualize FBE's performance, it was benchmarked against three well-regarded classifiers: *k*-Nearest Neighbors (KNN), Random Forest (RF), and AdaBoost. These classifiers were selected due to their popularity and proven track records in both academic and industrial settings, serving as a robust baseline for comparison. The parameter configurations for the experiments were carefully chosen to balance between model complexity and predictive performance:

- KNN and FBE: Configured with three nearest neighbors (*k*=3), a standard setting for KNN that offers a balance between underfitting and overfitting. For FBE, additional parameters included three iterations (*m*=3) to test the ensemble effect and a grace parameter (*g*=0) to evaluate its impact on model sensitivity and specificity.

- Random Forest: Utilized 100 fully grown decision trees to maximize the ensemble effect, enhancing the model's ability to generalize across different datasets.
- AdaBoost: Similar to RF in the number of decision trees but with trees pruned at level one to focus on reducing overfitting, enhancing the model's generalizability.

3) Performance evaluation metrics: Comprehensive metrics were selected to evaluate the performance of the models under test comprehensively:

- Accuracy: Provides a general measure of model correctness across all classes, useful for initial assessments of model efficacy.
- Precision: Critical for applications where the cost of a false positive is significant, helping to measure the reliability of the positive predictions.
- Recall: Especially important in medical or financial applications where failing to detect positives can have serious consequences, it measures the model's ability to capture all relevant instances.
- F1 Score: Combines precision and recall into a single metric that quantifies a model's accuracy at identifying only relevant instances, which is crucial for evaluating performance in imbalanced datasets.

4) Evaluation protocols: To ensure a thorough evaluation of FBE and to benchmark its performance against conventional models, we employ a robust cross-validation methodology. Specifically, the data is split in a 7:3 ratio, with 70% used for training the models and the remaining 30% dedicated to testing. This widely accepted split ratio allows for substantial training data while providing enough test data to assess model generalization effectively. Furthermore, the cross-validation process is repeated 10 times to ensure the reliability and stability of the performance metrics. Each iteration randomly redistributes the data according to the 7:3 training-to-testing ratio, minimizing bias and variability in the evaluation. The performance metrics are calculated for each run, and the results are then averaged across all 10 iterations to produce a final performance measure.

C. *Performance Analysis and Comparisons*

The experimental results are thoroughly detailed in Tables III and IV. Visual representations of these results are presented in Fig. 3 and 4.

Table III and Fig. 3 show that FBE consistently achieves high accuracy across all tested datasets, with standout performances on datasets like Iris (97%), Fertility (95%), and MNIST (80%). These results indicate a strong ability of FBE to handle both simple and complex data structures. On average, FBE achieves an accuracy of 87%, which is 7% higher than KNN and 11% higher than AdaBoost, and closely trails RF by only 1%. This demonstrates that FBE provides a robust alternative to more established models, particularly in handling varied data types effectively.

TABLE. III. ACCURACY AND F1 SCORE PERFORMANCE OF COMPARED MODELS ON VARIOUS DATASETS

ID.	Dataset	Accuracy				F1 Score			
		AdaBoost	*KNN*	*RF*	*FBE*	*AdaBoost*	*KNN*	*RF*	*FBE*
1	Breast Cancer	0.96	0.94	0.96	0.94	0.95	0.90	0.94	0.93
2	Lymphography	0.66	0.76	0.84	0.86	0.62	0.72	0.74	0.86
3	Fertility	0.80	0.85	0.88	0.95	0.56	0.59	0.64	0.49
4	ECG	0.86	0.96	0.97	0.91	0.51	0.87	0.87	0.54
5	Diabetes	0.76	0.69	0.76	0.79	0.73	0.66	0.72	0.77
6	Iris	0.94	0.95	0.96	0.97	0.93	0.94	0.95	0.97
7	Spambase	0.94	0.81	0.96	0.90	0.92	0.80	0.95	0.90
8	MNIST	0.35	0.55	0.75	0.80	0.29	0.46	0.44	0.66
9	Glass	0.49	0.70	0.80	0.77	0.35	0.61	0.69	0.78
10	Magic	0.84	0.80	0.88	0.85	0.82	0.76	0.87	0.66
Average		0.76	0.80	0.88	0.87	0.67	0.73	0.78	0.76

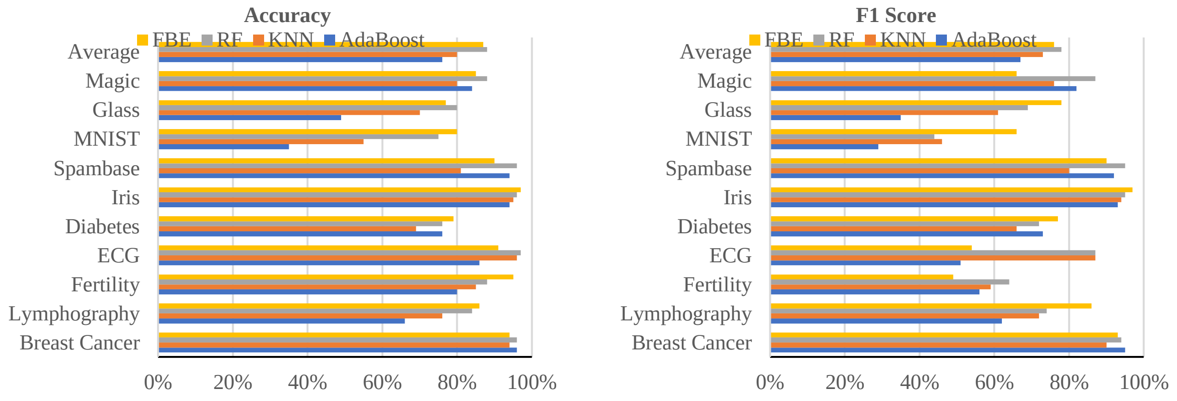


Fig. 3. Accuracy and F1 Score performance of compared models across various datasets

These tables and figures also illustrate how the F1 Score highlights FBE's balanced performance in precision and recall. Notably, FBE achieves a score of 78% on the Glass dataset, significantly outperforming AdaBoost's 35% and KNN's 61%. On the Iris dataset, it attains an impressive 97%. These results underscore the model's effectiveness in scenarios where balancing false positives and false negatives is crucial. With an average F1 score of 76%, FBE surpasses both KNN and AdaBoost, demonstrating its superior ability to harmonize recall and precision across diverse applications.

In terms of precision, as outlined in Table IV and Fig. 4, FBE shows exemplary results, especially in datasets like Iris (97%) and Fertility (95%), where accuracy in the positive predictive value is critical. FBE's average precision across all datasets stands at 86%, higher than both AdaBoost and KNN, underscoring its reliability in classifying instances correctly.

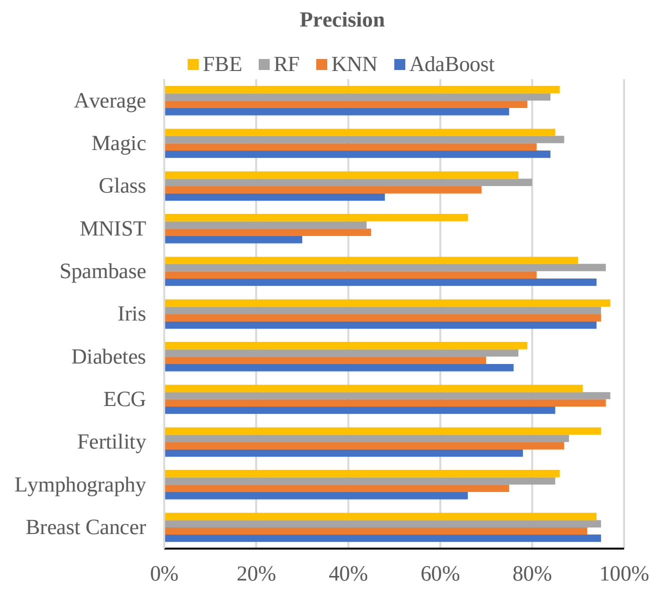
For the recall metric, Table IV and Fig. 4 reveal FBE's strength in sensitivity, particularly notable in datasets like ECG (85%) and Glass (79%). It maintains an average recall of 79%, indicating its effectiveness in identifying all relevant instances across varied datasets. This capability is crucial for applications where missing an instance can have significant repercussions.

The comparative analysis reveals that FBE not only competes closely with, but in many cases outperforms, traditional models. This is particularly evident in its consistent superiority over AdaBoost and frequent outperformance of KNN. While RF often shows slightly higher metrics, the gap is marginal, suggesting that FBE can offer comparable performance with added benefits of efficiency in processing and model simplicity.

FBE's effectiveness can be attributed to its innovative approach in handling datasets. By focusing on the most informative features through its sorted subset and ensemble methods, it reduces the impact of noisy or irrelevant features that typically affect KNN algorithms. This feature prioritization not only enhances accuracy but also improves the model's ability to generalize across different data types, avoiding the overfitting commonly seen in traditional models.

TABLE. IV. PRECISION AND RECALL PERFORMANCE OF COMPARED MODELS ON VARIOUS DATASETS

ID.	Dataset	Precision				Recall			
		AdaBoost	***KNN***	***RF***	***FBE***	***AdaBoost***	***KNN***	***RF***	***FBE***
1	Breast Cancer	0.95	0.92	0.95	0.94	0.95	0.91	0.94	0.94
2	Lymphography	0.66	0.75	0.85	0.86	0.63	0.72	0.75	0.84
3	Fertility	0.78	0.87	0.88	0.95	0.56	0.62	0.65	0.48
4	ECG	0.85	0.96	0.97	0.91	0.52	0.83	0.80	0.85
5	Diabetes	0.76	0.70	0.77	0.79	0.71	0.65	0.73	0.76
6	Iris	0.94	0.95	0.95	0.97	0.92	0.95	0.95	0.97
7	Spambase	0.94	0.81	0.96	0.90	0.92	0.80	0.93	0.90
8	MNIST	0.30	0.45	0.44	0.66	0.42	0.50	0.45	0.74
9	Glass	0.48	0.69	0.80	0.77	0.41	0.61	0.69	0.79
10	Magic	0.84	0.81	0.87	0.85	0.82	0.76	0.86	0.63
Average		0.75	0.79	0.84	0.86	0.69	0.74	0.78	0.79



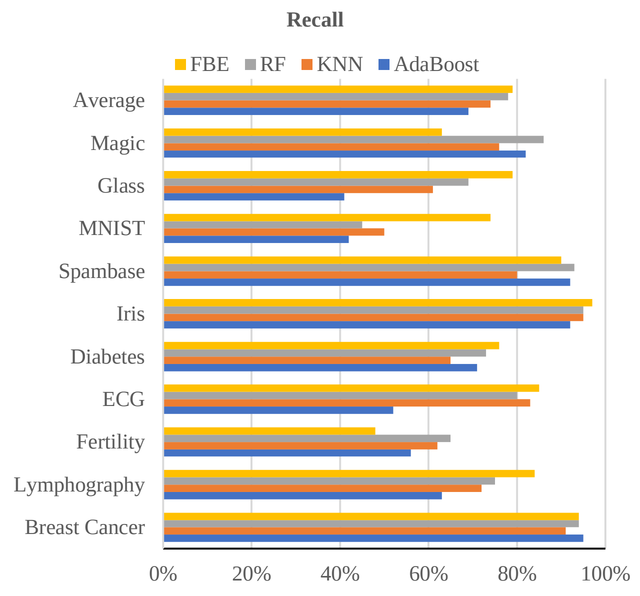


Fig. 4. Precision and Recall performance of compared models across various datasets

The reasons behind FBE's enhanced performance are manifold:

- Ensemble advantage: FBE uses multiple sorted subsets, reducing variance and improving reliability through ensemble averaging. This approach mitigates the impact of outlier data points and feature noise, which can significantly affect models like KNN and AdaBoost.

- Feature selection: By iteratively focusing on the most informative features, FBE minimizes the challenges of dimensionality and irrelevant feature noise, a frequent issue in high-dimensional datasets like MNIST.

In summary, the detailed results highlight the high potential of FBE for handling high-dimensional or imbalanced data. Its performance across all metrics demonstrates both robustness and adaptability in addressing various classification challenges. The comprehensive evaluation of FBE against standard models reveals its effectiveness and versatility across diverse datasets. Its systematic feature selection and use of ensemble methods enhance its accuracy and reliability in complex classification tasks, spanning fields from healthcare to image processing.

V. Conclusion and Future Directions

In this research, we have introduced a robust ensemble algorithm aimed at enhancing the performance of *k*-nearest neighbors classification through the innovative use of feature bagging. Our method involves selecting a subset of features, determining the most informative feature within this subset using the mutual information metric, and utilizing this feature to sort the data subset. This sorting facilitates an efficient binary search during the testing phase to quickly locate approximate nearest neighbors, and the process is iterated multiple times to improve classification performance and reliability. The proposed algorithm also undergoes a rigorous complexity analysis in both the training and testing phases. This analysis confirms that our approach not only improves performance metrics but does so with a significant reduction in computational overhead, moving from linear to logarithmic complexity.

Our experimental validation shows that the proposed algorithm significantly outperforms traditional *k*-nearest neighbors and AdaBoost in terms of accuracy, precision, recall, and F1 score across various datasets, including those with high-dimensional and imbalanced data. Notably, our approach shows marked improvement on datasets like MNIST, where traditional *k*-nearest neighbors typically struggle due to the curse of dimensionality and noise sensitivity. The ensemble algorithm consistently achieves higher accuracy rates, often exceeding the performance of the Random Forest in specific scenarios, particularly with imbalanced datasets.

Future research will expand the robustness studies of our algorithm across a broader range of datasets, especially exploring its performance under extreme conditions of data distribution and class imbalance. This research paves the way for future studies to explore hybrid approaches that combine feature bagging with other machine learning techniques to further enhance classification performance and computational efficiency.

Acknowledgment

This work has been supported by the College of Information Technology and Communication (CICT) at Can Tho University. We extend our heartfelt thanks to the Big Data and Mobile Computing Laboratory of CICT for their invaluable assistance. Additionally, we received support from the European Union's Horizon research and innovation program under the MSCA-SE (Marie Skłodowska-Curie Actions Staff Exchange) grant agreement 101086252; Call: HORIZON-MSCA-2021-SE-01; Project title: STARWARS (Stormwater and Wastewater Networks Heterogeneous Data AI-Driven Management).

References

[1] M. Magris and A. Iosifidis, "Bayesian learning for neural networks: an algorithmic survey", Artificial Intelligence Review, 56.10 (2023): 11773-11823, 2023.

[2] I.H. Sarker, "Machine learning: Algorithms, real-world applications and research directions", Springer Nature Computer Science, 2.3 (2021): 160, 2021.

[3] Y. Eren and I. Kucukdemiral, "A comprehensive review on deep learning approaches for short-term load forecasting", Renewable and Sustainable Energy Reviews, 189 (2024): 114031, 2024.

[4] P. Cunningham and S.J. Delany, "*k*-nearest neighbour classifiers - a tutorial", ACM Computing Surveys (CSUR), 54.6, pp. 1-25, 2021.

[5] A. Shokrzade, M. Ramezani, F.A. Tab and M.A. Mohammad, "A novel extreme learning machine based knn classification method for dealing with big data", Expert Systems with Applications, 183, 115293 (2021).

[6] S. Shekhar, N. Hoque and D.K. Bhattacharyya, "Pknn-mifs: A parallel knn classifier over an optimal subset of features", Intelligent Systems with Applications, 14, 200073 (2022).

[7] P. Gupta, A. Jindal, Jayadeva and S. Debarka, "Combi: Compressed binary search tree for approximate k-nn searches in hamming space", Big Data Research, 25, 100223 (2021).

[8] A.B. Hassanat, "Norm-based binary search trees for speeding up knn big data classification", Computers, 7(4), 54 (2018).

[9] P. Pappula, "A novel binary search tree method to find an item using scaling", International Arab Journal of Information Technology, 19(5), pp. 713-720, 2022.

[10] H. Saadatfar, S. Khosravi, J.H. Joloudari, A. Mosavi, S. Shamshirband, "A new *k*-nearest neighbors classifier for big data based on efficient data pruning", Mathe- matics 8(2), 286, 2020.

[11] H. Wang, P. Xu and J. Zhao, "Improved knn algorithms of spherical regions based on clustering and region division", Alexandria Engineering Journal 61(5), pp. 3571–3585, 2022.

[12] N. Islam, M. Fatema-Tuj-Jahra, M.T. Hasan and D.M. Farid, "Knntree: A new method to ameliorate *k*-nearest neighbour classification using decision tree", In International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1–6, IEEE, 2023.