Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.12 n.58, p. 589-601. March/April, 2026.

DOI: https://doi.org/10.5935/jetia.v12i58.3192

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

DESIGN OF A HYBRID ENSEMBLE FEATURE SELECTION FRAMEWORK FOR BIG DATA TEXT MINING

Smari Samah*[1], Barigou Fatiha[2] and Belalem Ghalem[3]

[1, 2, 3]Computer Science Department, Laboratory of Computer science of Oran (LIO), Oran 1 University, Ahmed Ben Bella, Oran, Algeria.

[1]https://orcid.org/0000-0002-5787-5380, [2]https://orcid.org/0000-0001-5444-4000, [3]https://orcid.org/0000-0002-9694-7586

Email: *smarisamah@gmail.com, fatbarigou@hotmail.com, belalem.ghalem@univ-oran1.dz

ARTICLE INFO

Article History
Received: December 24, 2025
Reviewed: January 1, 2026
Accepted: January 16, 2026
Published: March 31, 2026

Keywords:
Approximate Computing,
Big Data,
Ensemble Feature Selection,
Text Mining,
Distributed Computing.

ABSTRACT

The growing volume of textual data often exceeds the capacity of available computing resources, and conventional machine learning algorithms struggle to scale up. Today, the quality of data is becoming more critical than its raw quantity: it is therefore essential to transform massive data into intelligent data through appropriate pre-processing steps. Feature selection plays a key role in this process. In this work, we propose the design of a hybrid ensemble-based feature selection framework for processing large-scale textual data. The approach is based on the MFD-AFSA algorithm combined with different feature evaluation functions, applied on multiple data subsets. To improve scalability, we also outline a distributed strategy in an Apache Spark environment, based on the Random Sample Partitioning model. Finally, we introduce an automatic approximation mechanism, which we call auto-approximation, enabling selection sets to be built dynamically via an approximation technique. This work is part of a methodological design approach; experimental validation and practical evaluations will be the subject of future work.

Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

The extensive and intensive use of communication tools everywhere produce huge amounts of data especially unstructured ones such as text, creating "Big textual Data" [1], indeed it is now more challenging to reveal useful insights in high dimensional unstructured data [2]. As a result, there has been a significant focus on methods and techniques used in text mining (TM) in order to be able to automatically extract useful information from this big textual data. In short, text mining as a process relies on a number of techniques such as textual data pre-processing, machine learning, etc. to extract high-quality information from text [3]. Recently, an up-trending concept has been introduced on how to operate a transition between Big Data to Smart Data [4], the latter focuses on the veracity and value of data, aiming to highlight the valuable data by filtering out all the imperfections and noise accumulation in high dimensional and large sample size datasets [5]. Regarding text mining, Big Data preprocessing is an essential way to achieve Smart Data. Yet this preprocessing step is heavily affected by the dimension explosion, this phenomenon motivates the dimensionality reduction of the overwhelming size of features contained in textual documents.

In such a case, only an optimal set of relevant features is considered during the learning phase. Although feature selection [6-8] is presented as a classic remedy for large-scale learning methods, this task is necessary to deal with the combinatorial explosion of the big dimension that prevents its functioning on Big Data, but also, to find the most compact and informative set of features in corpora with high degree of redundancy and irrelevance. Thus, Big Data dimensionality reduction needs new algorithms with low cost linear or sublinear runtimes while maintaining the required performance [9]. In [10] authors show that there is a gap between the growing dimensionality and the feature selection (FS) algorithms capacity in handling such data, because most feature selection algorithms lack complex requirements, such as processing speed and storage reduction in their design. Distributed and parallel computing frameworks are essential to design and implement scalable and effective feature selection, nevertheless traditional centralized text feature selection cannot be exploited directly, as it may not scale well enough to tackle this large number of features.

In fact, in distributed environment, the performance of FS methods deteriorates as the number of nodes increases, due to greater information loss [11]. Moreover, developing a parallel algorithm from scratch is often challenging and does not necessarily guarantee optimal performance. For example, subset-based selectors rely on pairwise feature correlations; in a parallel setting, this requires computing interactions between millions of dimensions distributed across different data locations, which is impractical. Another important factor to consider is the changing characteristics in Big Data, the high sensitivity of the feature selection outcomes to the slightest changes in input data i.e. algorithmic instability, affect the reliability of the learning process. This is particularly critical in contexts where data is dynamic and constantly evolving, necessitating robust feature selection methods that can adapt to these fluctuations [12]. Hence, Large-scale feature selection can be addressed from two perspectives: either by designing novel sequential methods that explicitly address the above requirements [13] or by developing distributed solutions inspired by standard methods [14, [15]. However, both approaches are non-trivial, as they involve significant challenges such as redesigning core algorithmic operations or managing the complexities and potential drawbacks of distributed implementations.

Ensemble feature selection methods have shown promising results for text classification, especially in managing high-dimensional and imbalanced datasets. These methods aim to increase the robustness and stability of feature selection by combining multiple selectors. This can be achieved either by using the same algorithm on different data subsets (data-centered ensembles) or by combining different algorithms (model-centered ensembles) [16]. Homogeneous, heterogeneous, and hybrid ensemble strategies have been explored, often relying on rank aggregation techniques to merge feature rankings and enhance classification performance [17]. Furthermore, ensemble feature selection is generally more robust to noisy data and offers better stability than individual feature selection algorithms when dealing with high-dimensional inputs [18]. In addition, distributed ensemble feature selection approaches can optimize computation time by subsampling the full feature space into manageable subsets. This reduces algorithmic complexity and improves classification accuracy without the need to explicitly design parallel feature selection algorithms. Although the ensemble feature selection is intrinsically amenable to a distributed implementation, the adaptation of a novel ensemble feature selection to Big textual Data scenarios is still an ongoing process, studies that permit an ensemble evaluation based on scalability, accuracy, diversity and stability are lacking in the scientific literature [19].

Basically, there are two main approaches to achieving parallelism in ensemble feature selection: (1) applying different feature selection models independently to the same big dataset, or (2) applying a single feature selection model to multiple training subsets derived from the original dataset. In terms of scalability each model has its pros and cons, for the former, analyzing a big data set at once by each method may require more resources than the ones available in order to build an ensemble feature selection model, though needless to parallelize the feature selection and thus avoiding the superfluous communications within the computing cluster. The second model is more suitable to large-scale feature selection since each model is constructed on multiple subsets of the big data which alleviates the complexity and the size of the dataset in timely manner, noteworthy the number and the size of the optimal training sets for ensemble feature selection is still debatable [20], [21]. The computational cost of an ensemble method depends on the number of its components, i.e., the data subsets, which are generated using a sampling technique such as the bootstrap resampling scheme [22]. This approach involves creating multiple samples–with replacement–from the original dataset, each having the same size as the original. Nonetheless applying bootstrapping on Big Data is computationally intractable, as an alternative, authors in [23] introduced the Bag of Little Bootstraps (BLB) which performs bootstraps on smaller samples of data then combines the results, the efficacy of BLB was demonstrated in constructing ensembles of classifiers in [24], where dramatic gain in training time was achieved.

Even with the advantages of BLB, the whole distributed big dataset needs to be loaded and scanned each time a small sample is drawn, rendering the random sampling process in computing clusters prohibitively expensive. An especially relevant question in the context of Big Data is whether ensemble feature selection requires processing the entire dataset, or whether comparable results can be achieved using only a limited number of data subsets. According to [25], using the full dataset offers little additional efficiency, as a small set of representative subsets may be sufficient to produce approximate yet reliable results. This idea aligns with the principles of approximate computing [26], a promising approach aimed at reducing resource consumption and computation time in cluster-based environments by trading exactness for efficiency. To address the challenges associated with large-scale text data sampling, feature selection can be reformulated as an asymptotic ensemble approach. In this way, incremental processing is applied to small representative subsets of data in order to gradually improve the selected features. The aim is to get a robust and reliable set that closely matches the results from the whole dataset while reducing computational costs. Considering these factors, our contribution consists of the conceptual design of the following essential components:

i. In order to improve the stability and diversity of the selected textual features, a hybrid textual feature selection framework (HTFSE) is designed by diversifying the data and feature selection techniques.
ii. An ensemble strategy of filter-based feature selection techniques is proposed, where words are ranked using multiple and heterogeneous feature evaluation functions (FEFs), without relying on a predefined cut-off threshold for subset selection.
iii. A distributed and parallelizable architecture for the framework HTFSE by using Apache Spark to enable efficient processing of large-scale text data.
iv. An integration of approximate computing into the ensemble process, combined with the Random Sample Partition (RSP) model, to reduce computational cost while preserving approximation quality.
v. A conceptual model for automatic hyperparameter adjustment, referred to as Auto-Approximation, designed to construct ensemble approximations adaptively and efficiently.

The remaining sections of this paper are organized as follows. Section 2 reviews recent research related to feature selection and ensemble learning methods in the context of Big Data text mining. Section 3 presents the proposed conceptual framework a hybrid ensemble model that leverages both data perturbation and function perturbation to increase diversity at multiple levels. The framework's elements and architectural integration into a distributed Apache Spark environment are also described in this section.

In Section 4, the theoretical contributions and implications of the proposed approach, as well as its future implementation, are examined. Finally, Section 5 concludes the paper and outlines directions for further work, particularly in terms of its implementation.

II. RELATED WORK

Text classification depends a lot on feature selection (FS), especially in Big Data environments where high dimensionality and data volume make performance and scalability a challenge. Despite their simplicity, traditional FS techniques such as TF-IDF, Mutual Information (MI), Information Gain (IG), and Chi-Square (CHI) are commonly used [6]. But, in large, dispersed, and unbalanced textual datasets, they frequently fall short in capturing deeper semantic relationships or scaling effectively. Filters, wrappers, embedded approaches, and whether they rely on subset evaluation or individual feature ranking are the usual categories into which classical FS methods are divided [6]. Because of their independence from particular classifiers, speed, and algorithmic simplicity, filter-based approaches are the most appropriate for large-scale settings [27]. However, the exponential growth of data in both volume and dimensionality has exposed the limitations of traditional FS techniques, which often suffer from memory bottlenecks and high computational costs. That's what led to the development of scalable FS solutions designed for parallel and distributed computing frameworks like Hadoop and Spark. As an example, [28] proposes DiReliefF, a distributed version of the ReliefF algorithm, implemented in Apache Spark to efficiently process large-scale datasets.

The algorithm keeps the accuracy of the original ReliefF while improving scalability and processing time a lot, which makes it useful for Big Data applications. Similarly, [29] introduces RDTFD, a Particle Swarm Optimization (PSO)-based FS method tailored to text classification. While it improves accuracy over traditional filters, it leads to greater runtime and has not been parallelized, raising questions about its applicability to large-scale datasets. A comprehensive overview of FS in Big Data is given by the survey in [30], which also highlights enduring issues like data heterogeneity, scalability, and real-time constraints. It also encourages parallel and cloud-based adaptations across FS paradigms. Textual data presents even more pronounced challenges due to its high dimensionality, sparsity, and semantic variability. Traditional FS algorithms become inefficient or ineffective in such settings. In response, several methods have been designed specifically for large-scale text. In [31], a MapReduce-based implementation of TF-IDF was introduced to accelerate Naïve Bayes classification. Although it showed improved efficiency, the system was deployed on a single node, limiting knowledge of its scalability. Using max and mean weighting, GTFIDF [32], which is deployed on Apache Hadoop, suggests a global TF-IDF approach. On large corpora, it performs better than conventional filter methods; however, it does not assess the effect of scaling node clusters.

All documents in a category are combined into a single document using CTF-CHI² [33], a MapReduce-based combination of Chi-Square and category-term frequency for multi-label classification. Although correlation modeling is improved, there is a chance that term importance will be distorted, and feature thresholding and scalability are not well understood. Hybrid approaches like MTF-MI (Maximum Term Frequency–Mutual Information), which are implemented in Hadoop, are presented in other studies, such as [34]. It demonstrates the well-known trade-off between relevance and scalability by outperforming CHI, MI, and TF-IDF in F- measure performance but degrading performance as feature count rises. Ontology-guided methods such as Taxonomic Feature Selection [35] address semantic limitations by organizing terms hierarchically. Though effective in generalization, these methods require curated taxonomies, reducing portability across domains. Additional advancements include frequency-discriminative methods for distributed FS implemented on Apache Spark, such as ALOFT, MFD, MFDR, and cMFDR [36]. Features are ranked by ALOFT according to local significance and document frequency. To refine feature sets, MFD and its variations use class-aware filtering, redundancy reduction, and mutual information. Although these methods are more appropriate for large textual datasets, they still depend on statistical heuristics, which might ignore semantic or contextual richness.

Recently, new FS paradigms that transcend frequency-based heuristics have been inspired by large language models (LLMs). The works in [37] suggest text-based and data-driven FS strategies in which features are chosen based on attention patterns or embeddings from models such as LLaMA-2, ChatGPT, and GPT-4. Although these techniques improve robustness and semantic interpretability, they are computationally demanding and do not have native parallelism. Moreover, scalability depends on access to high-performance GPU clusters, and effectiveness hinges on the quality of textual descriptions, posing reproducibility and cost barriers. Furthermore, reproducibility and cost barriers arise because efficacy depends on the quality of textual descriptions, while scalability depends on access to high-performance GPU clusters. Ensemble feature selection has also been explored to improve stability, robustness, and classification performance in large-scale settings. For instance, [38] proposes an ensemble strategy by combining three baseline feature selection techniques: C4.5 decision tree, Principal Component Analysis (PCA), and Genetic Algorithm (GA), which stand for filter, wrapper, and embedded approaches, respectively. After comparing serial and parallel configurations, the study concludes that serial combinations produce feature sets that are more precise and condensed. It is noted that future research should focus on the stability of ensemble feature selection techniques, which have not yet been thoroughly investigated.

Furthermore, the suggested methods' applicability to Big Data contexts is limited because they are specifically tested on high-dimensional, low sample size datasets and fail to address the scalability problems that arise in large-scale datasets with both high dimensionality and a large number of instances. [39] suggests an ensemble of algorithms inspired by nature (OCFA, OCSA, OBBA, and MGWO), that improves performance but is unclear when it comes to aggregation and hasn't been tested on large datasets. [40] improves classification outcomes by combining Biogeography-Based Optimization (BBO) with Bagging; however, scalability has not been demonstrated. For social media sentiment analysis, [41] develops a Hybrid Ensemble Learning Model (HELM) which combines IG, CHI2, and AdaBoost; however, it ignores scalability and optimization.[42] suggests using Random Discretization (RD) together with PCA to create varied training subsets in order to address data imbalance. This enhances classifier diversity but suffers from long prediction times on imbalanced Big Data. DiCFS [43] implemented on Spark, introduces a distributed correlation-based FS (CFS) using horizontal and vertical data partitioning. It achieves strong performance but requires dataset-specific tuning and shows sensitivity to dimensionality. In [44], a new approach for selecting viral genomic sequence features is proposed. This approach, called PRCFX-DT, is based on probabilistic graph modeling to measure the importance of codons based on their centrality in the graph, which allows features to be extracted from genomic sequences.

This makes it possible to capture complex dependencies that conventional feature selection techniques might overlook. Although graph-based techniques have been developed for Big Data, their computational complexity remains very resource-intensive for very large data sets, requiring significant amounts of memory and processing power. Table 1 provides a comparative summary of the related works discussed above. Numerous distributed or parallel feature selection methods have been developed to address the challenges posed by high-dimensional data. While they generally save time and memory resources, they also have limitations. For example, DiReliefF, based on Spark, offers better performance than its MapReduce version, but it remains dependent on fixed data partitions and does not allow for incremental updates or intelligent aggregation of results between different subsets. Other approaches like those based on evolutionary algorithms give good results in terms of accuracy but are often computationally expensive and difficult to interpret.

More recent strategies, combining task parallelism or hybrid approaches, have been explored, but they rarely address essential aspects such as the stability and robustness of the selected feature, especially in the context of textual datasets, which are often massive and complex. Another common issue is the assumption that all data is available from the beginning, or the use of fixed partitions. This can lead to unreliable variable choices, especially when dealing with noisy, heterogeneous, or changing data. Very few studies focus on progressive or asymptotic strategies that are capable of learning gradually from representative samples while limiting computational costs. These limitations motivate our work, which aims to design ensemble feature selection in the context of Big Data as an asymptotic and incremental process, aligning distributed learning constraints and efficiency goals. In the next section, we detail the conceptual foundations and architecture of our proposed approach. Table 1 provides a comparative summary of the related works discussed above.

Table 1: Comparison of feature selection strategies in the context of Big Data.

Ref Year	FS technique	Type of data	Parallelization	Scalability	Limitations
[28] 2018	DiReliefF	Numerical & categorical	Data Parallel	a high degree of scalability for large datasets, especially when using horizontal partitioning	The potential inefficiency when dealing with extremely large and high-dimensional datasets.
[29] 2020	RDTFD (positive/negative discrimination), PSO optimization.	Textual Spam corpora	Parallel PSO considered (not implemented).	Medium to High	High time cost, limited generalizability to spam.
[31] 2018	TF-IDF	Textual	Data parallelism	Medium	Single node environment. Scalability not verified in multi-node environments
[32] 2021	GTFIDF	Textual	Data parallelism	Medium	Single node environment. Time cost of node increase not analyzed.
[33] 2020	CTF-CHI²	Textual	Data parallelism	Medium	Combining documents into one aggregated document per category can impact feature relevance
[34] 2021	MTF-MI	Textual	Data parallelism (Hadoop/MapReduce)	Medium	Becomes less performed for a given threshold of selected features.
[36] 2019	ALOFT, MFD, MFDR, cMFDR, AFSA for automation.	Textual	Moderate, as filters are parallelizable, but no distributed framework is mentioned	Moderate (scalable filter and AFSA, limited by lack of Big Data)	Limited generalizability across classifiers and feature evaluation criteria
[37] 2024	LLMs: GPT-4, ChatGPT, LLaMA-2	umerical & categorical (biological + clinical)	Not specified (LLM intrinsic capability).	scaling up LLM size enhances text-based methods more consistently	Performance scales with model size, but limited by compute and diminishing returns. LLMs struggle to handle high-dimensional inputs (e.g., genomic features) at scale.
[38] 2020	Parallel & serial combinations using PCA, GA, and C4.5 FS methods	Genomic Textual	No explicit parallel implementation tested	Potentially scalable via parallelism, but not tested on large-scale datasets. No data partitioning limits applicability in Big Data contexts.	Focuses on low-sample-size datasets despite the mention of big data; lacks data partitioning into subsets, preventing complexity reduction through distributed computation
[39] 2020	aggregation of OCSA, OCFA, OBBA, MGWO	Biomedical	Adaptable to parallel environments due to algorithm independence, but not implemented or tested on a cluster.	OCSA, OCFA, OBBA, and MGWO are wrapper- based methods requiring repeated classifier evaluations per feature subset, which limits scalability due to high computational cost on datasets with many features or large sample sizes.	Focused on high- dimensional, low-sample-size datasets; scalability to large datasets not assessed.
[40] 2020	BBO	Textual	Bagging enables potential parallelization, but no implementation (e.g., MapReduce/Spark) or distributed experiments are reported.	While the use of subsets via Bagging improves scalability for large datasets the wrapper-based nature of the BBO algorithm increases computational complexity. The scalability remains untested on truly large- scale (big	High computational complexity due to BBO (wrapper); lacks discussion on the number/size of data subsets used for Bagging.

				data) datasets.	
[41] 2020	Ensemble FS (IG and CHI²)	Textual	No explicit parallel implementation.	While filter methods (IG, CHI) are scalable for large textual datasets, the sequential nature of AdaBoost and the lack of experimentation on massive big data limit the overall scalability of the approach	Not well-suited for large corpora: the Twitter datasets tested are not explicitly described as massive
[42] 2021	Ensemble FS (RD and PCA)	Tabular	The solution is designed for big data scenarios and leverages the MapReduce paradigm to distribute computation across multiple nodes. Each decision tree is trained on an independent data subset, making the approach highly parallelizable	Highly scalable (Spark, distributed SMOTE), empirically validated in a big data context	Complexity of distributed SMOTE, reliance on decision trees, and lack of discussion on subset optimization
[43] 2019	DiCFS	Numerical	Two parallelization strategies: horizontal & vertical partitioning using Apache Spark	Highly scalable (Spark, distributed correlation calculations), tested	Quadratic complexity, partitioning choice data- dependent.
[44] 2025	PRCFX-DT	Genomic sequences	No explicit integration of parallelization strategies	Limited scalability for large-scale or high-dimensional data	Computational limitations and efficiency

Source: Authors, (2026).

III. DESIGN OF THE PROPOSED FRAMEWORK

We propose a framework that addresses the large model-to-large data challenge. The goal is to leverage the strengths of ensemble learning in a Big Data context without having to choose between scaling the model or scaling the data. The key idea is to explore how a powerful machine learning pipeline can be effectively applied to both the learning process and large-scale data. To achieve this, we introduce a flexible ensemble framework that combines data perturbation and model perturbation strategies to maximize diversity. This results in a novel hybrid ensemble feature selection approach tailored for big textual data. The design of the proposed framework follows a multi-stage process, as illustrated in Figure 1.

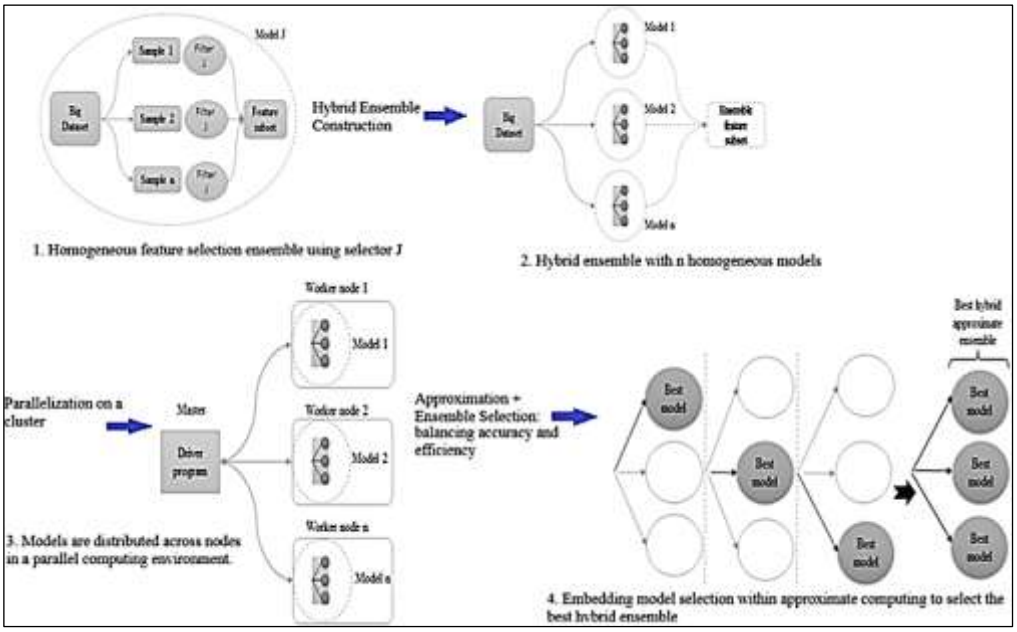

Figure 1: Variation of Spacing with Suspension Preload.
Source: Authors, (2026).

III.1 FILTER METHODS

Maximum Features per Document (MFD) [35], is a filter-based feature selection method that uses a Feature Evaluation Function (FEF) to rank features. It requires a parameter f, which represents the number of features to select per document. The method selects the top-f ranked features in each document to construct the final feature set. Unlike traditional FS methods that select m features from the entire corpus, MFD focuses on per- document selection. This strategy significantly reduces the search cost for the optimal f compared to the global optimization of m. To automate the choice of the f parameter, we use the Automatic Feature Subsets Analyzer (AFSA) [45], an auxiliary method designed to configure f dynamically. We use the combined $MFD - AFSA$ approach in the design of our ensemble system. This decision is driven by the fact that it does not depend on a predefined threshold and is suitable for environments with limited time and computational resources. Previous studies have demonstrated that MFD-AFSA provides a more efficient and scalable alternative in such contexts [36].

III.2 HYBRID TEXT FEATURE SELECTION ENSEMBLE

Feature selection ensemble techniques remain relatively underexplored in text mining (TM), especially compared to the widespread use of ensemble classifiers in sentiment or text classification tasks. However, the robustness (or stability) of the selected feature subset is as crucial as its optimality, particularly in knowledge discovery scenarios where interpretability and consistency are key. Two major ensemble strategies exist in feature selection:

- Homogeneous ensembles (data perturbation) [17]: These rely on resampling the dataset (e.g., via Bagging or Random Sampling [46–48]) to generate multiple randomized versions. The same feature selection method is then applied independently to each subset, improving robustness through repeated evaluations.
- Heterogeneous ensembles (function perturbation): These apply different FS algorithms (base selectors) to the same original dataset. By promoting functional diversity, they aim to produce a more reliable and domain-representative feature subset [17], [47].

Our approach was guided by the following finding: to design a holistic ensemble, we propose a hybrid ensemble that integrates multiple homogeneous ensembles into a heterogeneous one. The goal here is to inject diversity both at the data level (via partitioning) and the algorithmic level, in an effort to provide a trade-off between robustness and stability. As illustrated in Figure 2, the proposed Hybrid Text Feature Selection Ensemble (HTFSE) framework for Big Data is constructed through the following steps:

1. Preprocessing: Raw unstructured text is cleaned through tokenization, stop-word removal, and stemming across all documents in the corpus.
2. Representation: The cleaned corpus is transformed into a numerical document-word matrix using the Bag-of-Words (BoW) model.
3. Partitioning: The document-word matrix is randomly partitioned into k subsets, each referred to as a Matrix Block (MB). This strategy (detailed in Section III.4) simulates data perturbation for homogeneous ensembles.
4. Homogeneous Ensemble Construction: Each homogeneous component is built in parallel by randomly selecting one set of matrix blocks, e.g., MB1, MB9, MB15 and MB4, then the same Text Feature Ranker $TFRjj$ is applied to each matrix MB yielding different rankings.
5. Micro-Aggregation: The rankings produced by each homogeneous ensemble are aggregated using simple but effective rank fusion strategies: mean rank, geometric mean, and best rank, as formalized in Equations (1)-(3). These functions compute a global relevance score for each feature f_j , reflecting its importance across the ensemble of document subsets.

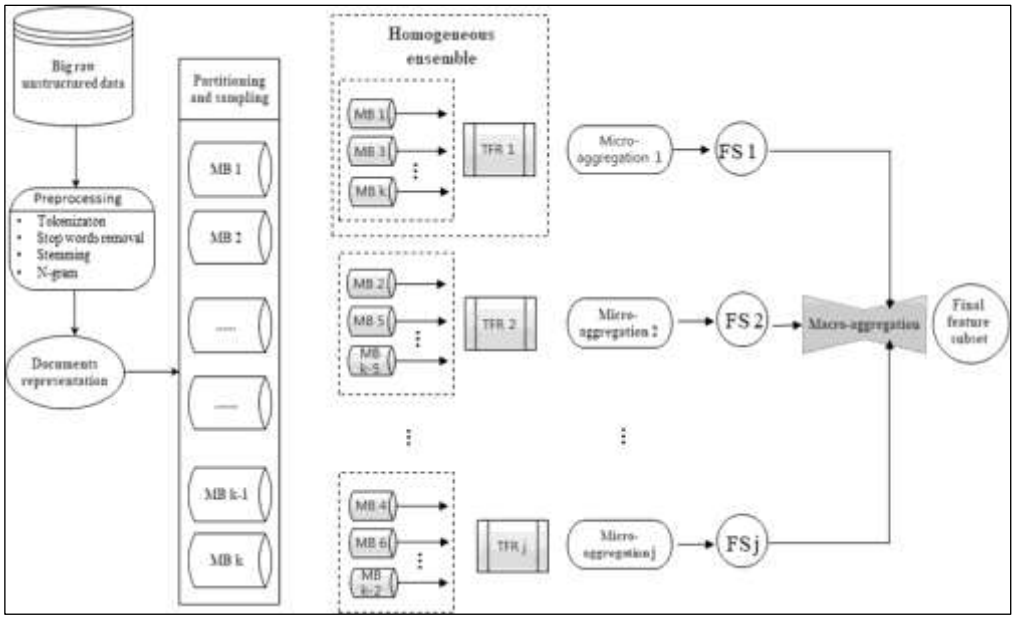

Figure 2: Overview of the hybrid text feature selection ensemble framework.
Source: Authors, (2026).

$$Mean = \frac{1}{I}\sum_{i=1}^{I} R_i(f_j) \quad (1)$$

$$Geom.mean = (\prod_{i=1}^{I} R_i(f_j))^{\frac{1}{I}} \quad (2)$$

$$Best.rank = Max\left\{R_i(f_j)\right\}_{i=1}^{I} \quad (3)$$

Where:

R is the relevance score of feature f.
I denotes all the positions that feature f has achieved among all rankings based on Text Feature Ranker TFRj.

1. The above-mentioned steps are repeated for each homogeneous ensemble for j different base selectors, generating intermediate feature subsets {FS1, FS2, FSj}.

2. Finally, the Macro-aggregation module a union- based combination of the diverse ensembles' outcomes is adopted trough equation (4) in order to estimate the finale subset of features.

$$Macro-agg(FS_1, FS_2, ..., FSj) = \bigcup_{j=1}^{J} FS_j \quad (4)$$

We apply the MFD-AFSA method across diverse Feature Evaluation Functions ($FEFs$). The pseudo-code of the hybrid $MFD-AFSA$ ensemble approach is presented in Algorithm 1.

Algorithm 1: Pseudo-code for the hybrid MFD-AFSA ensemble.

Input: *D, N, k, Q*
D//Pre-processed Dataset
N//The number of the Feature Evaluation Function (FEF)
Q //The number of matrix blocks
k// The number of matrix blocks in one batch
Output: EFS ensemble feature subset

```
1 :  Begin
2 :  S = Matrixblocks_Sampling (D,k)
3 :  For j=1 to N
4 :      For i=1 to k
5 :          FSi,j= Compute MFD-AFSA(MBi,FEFj)
6 :      End for
7 :      FSj=Micro_agg {MFD_AFSA(MBi)}        //i=1,..,k
8 :  End for
9 :      EFS= Macro_agg {FSj}                 //j=1,..,N
10 : End
```

Source: Authors, (2026).

The proposed algorithm is divided into three phases:

1. Document-term matrix sampling (line 2): The training matrix D is randomized and partitioned without replacement into QQ matrix blocks. A batch S is then created by randomly selecting k matrix blocks, where k < Q. This sample S is used to build homogeneous components.
2. Generation of intermediate feature subsets (lines 4-6): The same filtering method, denoted as $MFD-AFSA(FEFjj)$ is synchronously applied to each of the k matrix blocks in the sample. This result in k feature subsets for the $jjth$ Feature Evaluation Function (FEF). These subsets are then aggregated (line 7) to form an intermediate feature subset.
3. Construction of diverse feature models (line 3-8): Multiple filtering functions are applied independently to different sampled batches. For each FEF, a new matrix block sample is generated, leading to jj distinct intermediate subsets. These subsets are finally aggregated (line 9) through a macro-aggregation step to produce the final ensemble feature subset.

The next challenge is to choose and combine the best configurations in a scalable and computationally efficient way after the hybrid feature selection ensembles are built using MFD-AFSA and various FEFs. Due to time and resource limitations, processing the complete dataset is frequently impracticable in real-world big textual data scenarios. In order to automatically and effectively identify the most promising homogeneous ensembles while preserving high classification accuracy, the selection mechanism based on incremental approximate computing is introduced in the following section.

III.3 ALGORITHM SELECTION: AUTO-APPROXIMATION

This section presents our strategy for selecting and combining the most effective homogeneous ensemble models efficiently, with minimal computational cost and without sacrificing classification accuracy. The core idea is to construct a hybrid ensemble feature selection framework by combining the strongest homogeneous ensemble models. A versatile and efficient machine learning pipeline that can be adjusted to various datasets and feature evaluation functions (FEFs) is the goal of this hybrid approach. However, the required trade-offs between accuracy, training time, and resource consumption make it difficult to achieve such flexibility in the context of large-scale textual data. To address this, we employ incremental approximate computing within a distributed cluster environment. This strategy allows us to identify the most relevant data portions for feature selection, thereby approximating high-quality subsets that yield classification accuracy comparable to that achieved using the full dataset. A central issue we explore is the relationship between sample size, accuracy metrics, and aggregation strategies.

As illustrated in Figure 2, optimizing this framework involves tuning various big data hyperparameters –particularly those governing the structure and behavior of the hybrid feature selection ensembles. Our approach involves a greedy ensemble selection strategy [49] based on forward stepwise selection, applied to homogeneous feature selection ensembles constructed in parallel. These ensembles are designed by distributing the large-scale dataset across three different FEF-based pipelines. This strategy incrementally builds an ensemble, starting from an empty set and adding at each step the homogeneous ensemble that maximizes validation performance. In the proposed framework, we define three portfolios of homogeneous ensembles – each associated with a specific feature evaluation technique –by varying the number of sampled matrix blocks, the classifiers *Cf*, and the feature aggregation functions, as illustrated in Figure 3.

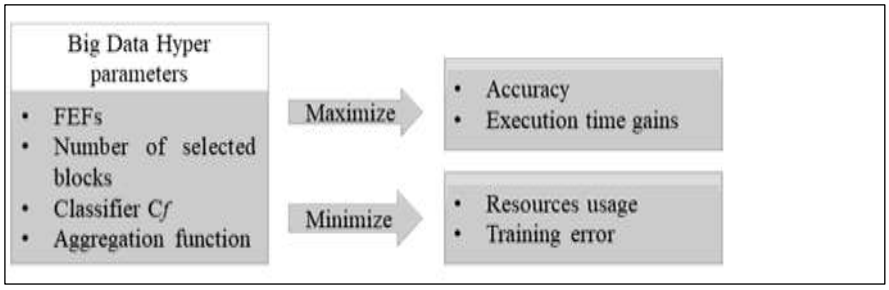

Figure 3: Hyperparameters tuning for Big Data hybrid ensemble.
Source: Authors, (2026).

Unlike the method in [43], the purpose here is not to combine different models to find the best performing ensemble, but we try to look out for the best configuration for each FEF ensemble library and then combine the three ensemble configurations into a hybrid ensemble. The selection of each base ensemble is based on the model averaging of four normalized performance metrics: the accuracy, the resources usage, training error and the execution time. Each measure is normalized to [0-1] scale and the model performance is converted to the mean normalized score across the four metrics. The superior base-level ensemble is the one that maximizes the performance measures for each FEF i.e. Max Avg {M (FEF)}. The approximate big data analysis via the Random Sample Partition RSP [50] discussed in section 3.4 , is a distributed data model that operates on randomized data blocks, is embedded within ensemble selection, this automatic ensemble construction is termed auto-approximation committed to a multiple Big Data hyperparameters setting. The homogeneous ensembles are built in parallel gradually by stepwise distribution of RSP blocks. As shown in Figure 4.

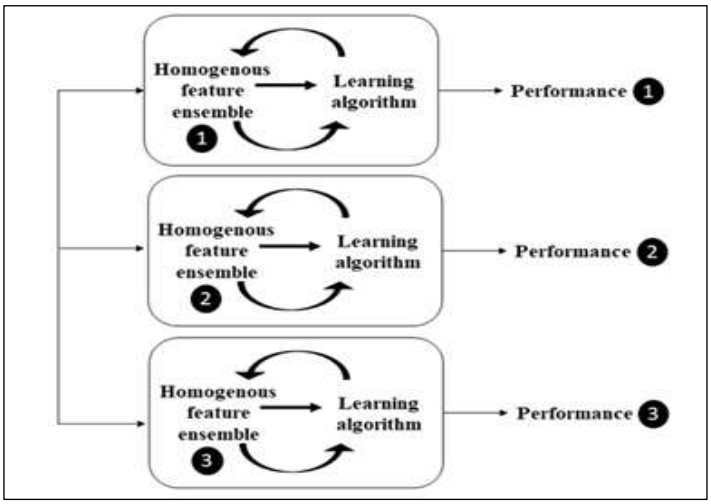

Figure 4: Parallel homogeneous ensemble library generation.
Source: Authors, (2026).

The auto-approximation strategy helps us establish the theoretical trade-offs between achieving satisfactory classification accuracy and the size of the random sample used. It also identifies the data synopsis - among the sampled RSP blocks - that is most relevant to the classification task. In addition, this approach enables the scalable construction of an optimal hybrid set (see Figure 5).

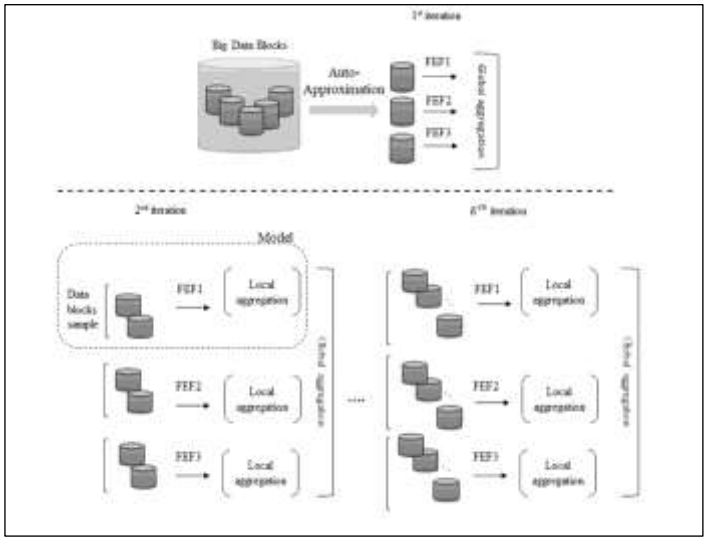

Figure 5: Auto-approximation ensemble analyses construction.
Source: Authors, (2026).

III.4 PARALLELIZATION OF THE PROPOSED FRAMEWORK HEFTS IN A SPARK CLOUD COMPUTING

Previous research can only be considered a first step towards a more profound understanding of the impactful option of the ensemble in text categorization, however scalability, speed up, stability and dataflow management of the ensemble in big text data, remains briefly addressed in the literature [48]. We proposed a multi virtual machines cluster parallelized with the Spark[i] framework and based on HDFS file system instead of using a cloud storage for better performance and for the data locality optimization (i.e. HDFS provides information about which nodes have copies of the data blocks, so the task scheduler optimize the tasks assignment), the cloud here was used as IaaS (Infrastructure as a Service) for the storage and computing purposes to build a scalable hybrid feature selection ensemble. The overall architecture of the system is shown in Figure 5. Firstly, we will plug the external library Natural Language ToolKit[ii] (NLTK) dependency to the Spark Driver, the input text files were stored to HDFS, and next Spark Driver accesses the underlying distributed file system to load the HDFS file, where a series of preprocessing methods are performed to generate the document-term matrix. Secondly the HDFS-file is partitioned and randomized through an algorithm that we will describe later. Usually when a file is loaded to HDFS, the file is sequentially partitioned into different blocks and replicated across the cluster nodes. The asymptotic ensemble model is particularly effective in Big Data scenario, as applying ensemble on the full feature space is less than optimal, so there is a need to create low dimensional feature subspace that keep the characteristics of the full space [45].

Sampling the training data in ensembles could be achieved through Bagging [51] which is a sampling with replacement of N instances of the original data, N is estimated about (63.2%) of the whole dataset [52], thus in case of a large dataset, Bagging is not suited as a sampling strategy. In the other hand Random Subsampling [53] is a good alternative to Bagging, however obtaining random samples of the huge document-term matrix requires a complete scan of the whole matrix each time a random sample is generated, such practice leads to a computationally expensive partitioning [54, 55]. In our Spark Framework we could have let HDFS manage the partitioning process through HDFS, nevertheless in these big data frameworks, the data partitions cannot be used as random samples, in fact Spark does not take into account the probability distribution of the data [56], this could lead to biased document classification. Hence to implement our Auto-Approximation framework, we adopted a two stage Random Sample Partition (RSP) technique: Data Chunking and Data Randomization, which can alleviate the problem of the unbalanced workload on different nodes due to the uneven distribution of the records, thus leading to tasks with different execution times [50]. The first stage operation is available in Apache Spark through range partition; the main characteristic of this strategy is that the statistical properties of the obtained blocks are similar to those of the entire dataset. Also, it is based on block-level sampling from an RSP model instead of record-level sampling (See Figure 6).

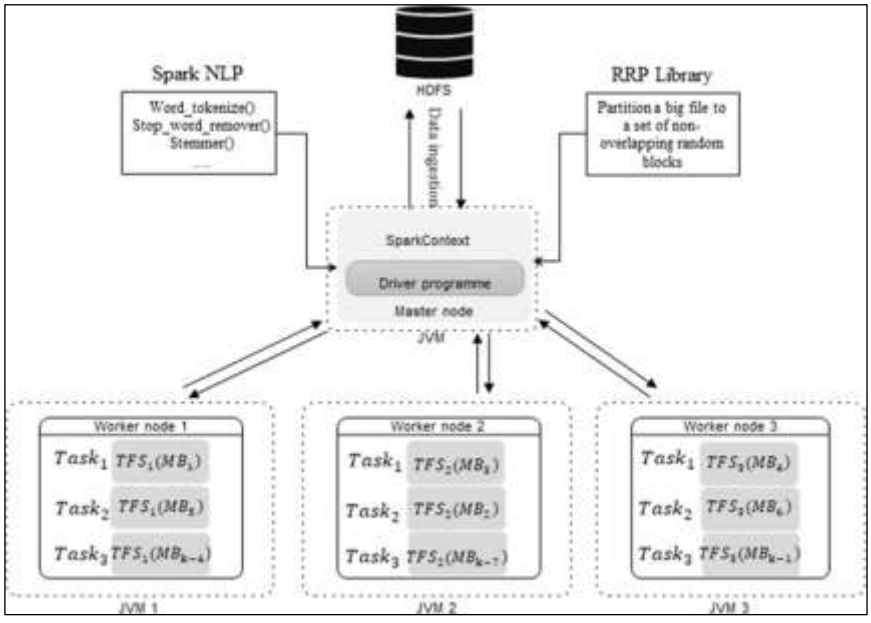

Figure 6: High-level architecture of spark inter-components interactions within cloud computing.
Source: Authors, (2026).

The document-term matrix is partitioned into *P* blocks, then stored to HDFS, consequently the matrices are converted to randomized blocks via the application of a RSP algorithm. The matrix block-level sampling consists of randomly withdrawing a batch of RSP blocks, note that the matrix blocks are generated in advance, instead of sampling a set of documents from the matrix file stored in HDFS. The HDFS matrix blocks are converted to an RSP model to obtain approximate results through round-random partitioning algorithm (RRP) [57], a horizontal partitioning scheme developed to overcome the computational cost of randomization in a previous RSP model [50], RRP assign a random document key K from RDD partition to a random block in a new RDD index calculated by K mod Q, then the resulting RRP blocks are distributed to one of Spark's executor cores (See Figure 7).

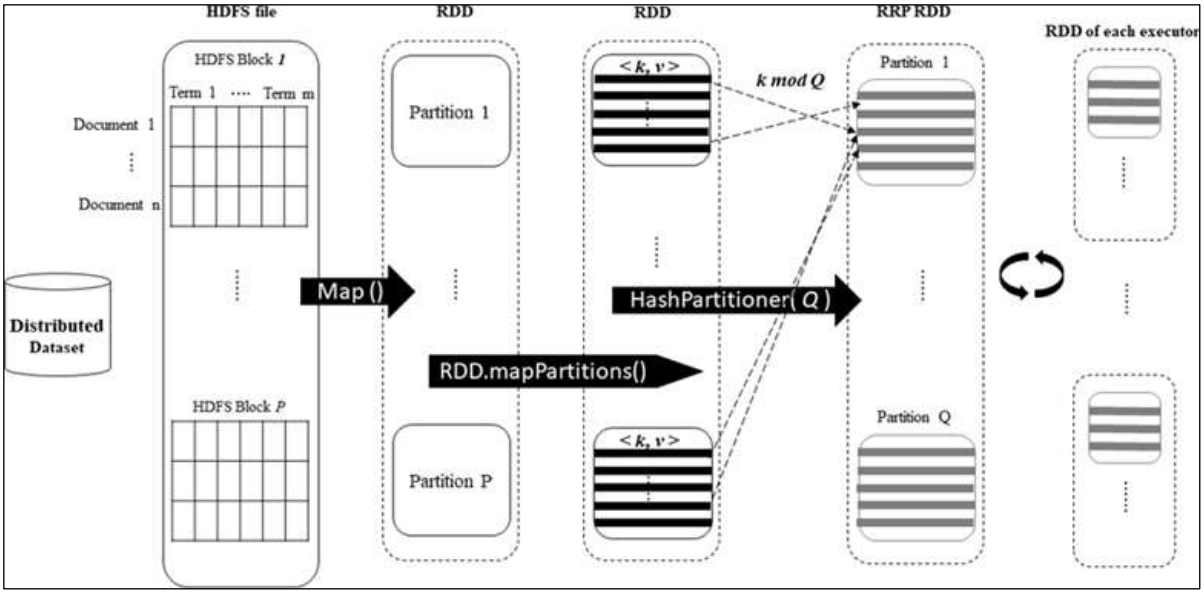

Figure 7: RSP modeling of the Documents representation using the RRP algorithm.
Source: Authors, (2026).

The reason for the text cleaning procedure before the partitioning and randomization process is that there no evidence that the same RSP data distribution is guaranteed compared to the big corpus, this automated framework not only permits to study the trade-off between the performance and RSP sample size, but it estimates how impactful would text transformations be on the representativeness power of RSP model.

IV. DISCUSSION

We have proposed a hybrid ensemble approach by decomposing the set of documents into a set of smaller matrices via a data approximation technique. This enables the feature selection algorithm to deal with totally random subsets of small size, guaranteeing both reduction of the search space (and hence complexity) and stability of textual feature selection. In the context of Big Data, there is no universal method for feature selection, and it is often necessary to combine different approaches to obtain a relevant subset. The model we have proposed combines an approximate data sampling technique with incremental distributed computation to guarantee both efficiency and optimality. In this section, we discuss the adaptations we have introduced to design a hybrid approach adapted to Big Data issues.

IV.1 ENSEMBLE TUNABILITY

Our hybrid ensemble's tuning approach is based on model-based optimization that has been specially modified for Big Data situations. Our tuning approach focuses on scalability and performance on large training sets, in contrast to traditional ensemble feature selection, which frequently emphasizes the number of feature selectors and pruning strategies. The hyperparameters we consider fall into two categories:

1. Functional hyperparameters, such as the selection function parameter f and the appropriate classifier for each Feature Ensemble Filter (FEF).
2. Data-level hyperparameters, notably the number of random document blocks selected for each homogeneous ensemble component, executed in parallel.

These hyperparameters are optimized to maximize performance gains in the hybrid ensemble while mitigating the cost of generalization performance evaluation.

IV.2 PARTITIONING

Creating homogeneous ensemble components is generally done through K-fold Cross-Validation (CV) splitting, CV in Spark depends upon the parallelism parameter which governs the maximum number of models to be trained at one, however building a set of ensembles with Spark-CV requires the whole training to be repeated by means of resampling to train the full training set, generating a large number of models which increases the risk of overfitting and the generalization performance of the hybrid ensemble is no longer effective, also it may exceed the available resources in the cluster. In the other hand it is highly time-consuming if the required random samples are generated by record-level sampling as it reads the big corpus document-by- document. To avoid this, our approach performs sampling at the RDD partition level, not at the record level. This strategy reduces data access overhead while maintaining representative sampling, significantly improving efficiency.

IV.3 AUTO-APPROXIMATE COMPUTING

A hybrid feature selection ensemble is intrinsically highly expensive to implement, let alone when ML pipeline is combined to Big Data. To address this, our framework introduces a dual strategy:

- Model-to-data fitting: The system automatically searches for the best ensemble configuration, independent of the feature selection function or dataset size.
- Data-to-model fitting: Only a few carefully chosen samples are used, rather than the entire dataset. These significantly cut down on computation time while roughly matching the full-scale model's performance.

This bidirectional adaptation merges approximate computing with ensemble selection, a concept we term "Auto-Approximation". This not only enhances the predictive power of the hybrid ensemble but also optimizes parallel computation costs.

V. CONCLUSION

In this work, we addressed the problem of feature selection in a Big Data context, proposing a hybrid and distributed approach adapted to the constraints of Big Data. We suggest a framework combining data approximation, incremental feature selection and distributed computing based on Apache Spark. It is based on an RDD-level partitioning strategy, hyperparameter optimization in a hybrid ensemble, and automatic configuration ("Auto-Approximation") of data blocks. While preserving adequate representativeness, this approach is designed to reduce computational complexity, increase the stability of selection results, and offer a better fit between the amount of data being examined and the resources at hand. Our future work will include a deep experimental evaluation on different datasets, as well as a systematic comparison with classical selection techniques in Spark MLlib. Other perspectives include the integration of an explicability layer to interpret selection criteria in the distributed environment

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Smari Samah.
Methodology: Smari Samah, Barigou Fatiha and Belalem Ghalem.
Investigation: Smari Samah and Barigou Fatiha.
Discussion of results: Smari Samah and Barigou Fatiha.
Writing – Original Draft: Smari Samah.
Writing – Review and Editing: Barigou Fatiha and Belalem Ghalem.
Resources: Smari Samah.
Supervision: Barigou Fatiha and Belalem Ghalem.
Approval of the final text: Smari Samah, Barigou Fatiha and Belalem Ghalem.

VII. REFERENCES

[1] M. Sigala, A. Beer, L. Hodgson, and A. O'Connor, "Big data for measuring the impact of tourism economic development programmes: A process and quality criteria framework for using big data," *in Big Data and Innovation in Tourism, Travel, and Hospitality*, M. Sigala, R. Rahimi, and M. Thelwall, Eds. Singapore: Springer, 2019, doi: 10.1007/978-981-13-6339-9_4.

[2] G. Nguyen et al., "Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z.

[3] F. Ayele, "Text mining technique for driving potentially valuable information from text," *Inf. Knowl. Manag.*, vol. 10, pp. 1–4, Jan. 2020.

[4] A. Luntovskyy, L. Globa, and B. Shubyn, "From big data to smart data: The most effective approaches for data analytics," *in Advances in Information and Communication Technology and Systems (MCT 2019)*, M. Ilchenko, L. Uryvsky, and L. Globa, Eds., Lecture Notes in Networks and Systems, vol. 152. Cham, Switzerland: Springer, 2021, doi: 10.1007/978-3-030-58359-0_2.

[5] S. Subha and J. G. R. Sathiaseelan, "Effective anomaly detection approach to classify noisy data using robust noise detection and removal technique in IoT healthcare data," *SN Comput. Sci.*, vol. 4, no. 5, Jun. 2023, doi: 10.1007/s42979-023-01890-2.

[6] X. Cheng, "A comprehensive study of feature selection techniques in machine learning models," Insights Comput. Signals Syst., vol. 1, no. 1, pp. 65–78, 2024, doi: 10.70088/xpf2b276.

[7] S. S. Subbiah and J. Chinnappan, "Opportunities and challenges of feature selection methods for high dimensional data: A review," *Ingénierie des Systèmes d'Information*, vol. 26, no. 1, pp. 67–77, 2021, doi: 10.18280/isi.260107.

[8] J. Zhang et al., "Feature selection for machine learning-driven accelerated discovery and optimization in emerging photovoltaics: A review," *Adv. Intell. Discov.*, vol. 2025, Art. no. 202500022, 2025, doi: 10.1002/aidi.202500022.

[9] F. Al-Turjman, H. Zahmatkesh, and L. Mostarda, "Quantifying uncertainty in internet of medical things and big-data services using intelligence and deep learning," *IEEE Access*, vol. 7, pp. 115749–115759, 2019, doi: 10.1109/ACCESS.2019.2931637.

[10] S. Kumar and M. Singh, "Big data analytics for healthcare industry: Impact, applications, and tools," *Big Data Min. Anal.*, vol. 2, no. 1, pp. 48–57, 2019, doi: 10.26599/BDMA.2018.9020031.

[11] L. M. Ang et al., "Deployment of IoV for smart cities: Applications, architecture, and challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2019, doi: 10.1109/ACCESS.2018.2887076.

[12] M. Firouznia, P. Ruiu, and G. A. Trunfio, "Robust feature selection for high-dimensional datasets using a GPU-accelerated ensemble of cooperative coevolutionary optimizers," *in Proc. 31st Euromicro Int. Conf. Parallel, Distributed and Network-Based Processing (PDP)*, Naples, Italy, 2023, pp. 291–298, doi: 10.1109/PDP59025.2023.00052.

[13] Y. Wu et al., "Large scale incremental learning," *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 374–382, doi: 10.1109/CVPR.2019.00046.

[14] A. Brankovic and L. Piroddi, “A distributed feature selection scheme with partial information sharing,” Mach. Learn., vol. 108, pp. 2009–2034, 2019, doi: 10.1007/s10994-019-05809-y.

[15] V. Palanisamy and R. Thirunavukarasu, “Implications of big data analytics in developing healthcare frameworks – A review,” *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 31, no. 4, pp. 415–425, 2019, doi: 10.1016/j.jksuci.2017.12.007.

[16] D. S. Guru et al., “Ensemble of feature selection methods for text classification: An analytical study,” *in Intelligent Systems Design and Applications (ISDA 2017)*, vol. 736. Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-319-76348-4_33.

[17] V. Bolón-Canedo and A. Alonso-Betanzos, “Ensembles for feature selection: A review and future trends,” *Inf. Fusion*, vol. 52, pp. 1–12, 2019, doi: 10.1016/j.inffus.2018.11.008.

[18] J. R. Saura, B. R. Herraez, and A. Reyes-Menendez, “Comparing a traditional approach for financial brand communication analysis with a big data analytics technique,” *IEEE Access*, vol. 7, pp. 37100–37108, 2019, doi: 10.1109/ACCESS.2019.2905301.

[19] R. Salman, A. Alzaatreh, and H. Sulieman, “The stability of different aggregation techniques in ensemble feature selection,” *J. Big Data*, vol. 9, no. 1, p. 51, 2022, doi: 10.1186/s40537-022-00607-1.

[20] C.-F. Tsai and Y.-T. Sung, “Ensemble feature selection in high dimension, low sample size datasets: Parallel and serial combination approaches,” *Knowl.-Based Syst.*, vol. 203, p. 106097, 2020, doi: 10.1016/j.knosys.2020.106097.

[21] C. Shang and F. You, “Data analytics and machine learning for smart process manufacturing,” *Engineering*, vol. 5, no. 6, pp. 1010–1016, 2019, doi: 10.1016/j.eng.2019.01.019.

[22] Y. Yu et al., “Clinical big data and deep learning: Applications, challenges, and future outlooks,” *Big Data Min. Anal.*, vol. 2, no. 4, pp. 288–305, 2019, doi: 10.26599/BDMA.2019.9020007.

[23] M. Huang et al., “A queuing delay utilization scheme for on-path service aggregation,” *IEEE Access*, vol. 7, pp. 23816–23833, 2019, doi: 10.1109/ACCESS.2019.2899402.

[24] G. Xu et al., “Internet of things in marine environment monitoring: A review,” *Sensors*, vol. 19, no. 7, pp. 1–21, 2019, doi: 10.3390/s19071711.

[25] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, “Smarter traffic prediction using big data, in-memory computing, deep learning and GPUs,” *Sensors*, vol. 19, no. 9, Art. no. 2206, 2019, doi: 10.3390/s19092206..

[26] S. Leonelli and N. Tempini, Eds., *Data Journeys in the Sciences*. Cambridge, MA, USA: MIT Press, 2020.

[27] C. Kuzudisli et al., “Review of feature selection approaches based on grouping of features,” *PeerJ*, vol. 11, p. e15666, 2023, doi: 10.7717/peerj.15666.

[28] R. J. Palma-Mendoza, D. Rodriguez, and L. De-Marcos, “Distributed ReliefF-based feature selection in Spark,” *Knowl. Inf. Syst.*, vol. 57, no. 1, pp. 1–20, Oct. 2018, doi: 10.1007/s10115-017-1145-y.

[29] Y. Liu, S. Ju, J. Wang, and C. Su, “A new feature selection method for text classification based on independent feature space search,” *Math. Probl. Eng.*, vol. 2020, Art. no. 6076272, pp. 1–14, 2020, doi: 10.1155/2020/6076272.

[30] M. Rong, D. Gong, and X. Gao, “Feature selection and its use in big data: Challenges, methods, and trends,” *IEEE Access*, vol. 7, pp. 19709–19725, 2019, doi: 10.1109/ACCESS.2019.2894366.

[31] H. Amazal, M. Ramdani, and M. Kissi, “A text classification approach using parallel Naive Bayes in big data context,” *in Proc. 12th Int. Conf. Intelligent Systems: Theories and Applications (SITA)*, New York, NY, USA: ACM, 2018, Art. no. 36, 6 pp., doi: 10.1145/3289402.3289536.

[32] H. Amazal, M. Ramdani, and M. Kissi, “A parallel global TF-IDF feature selection using Hadoop for big data text classification,” *in Advances on Smart and Soft Computing*, vol. 1188, F. Saeed et al., Eds. Singapore: Springer, 2021, pp. 107–117.

[33] H. Amazal, M. Ramdani, and M. Kissi, “Towards a feature selection for multi-label text classification in big data,” *in Smart Applications and Data Analysis*. Cham, Switzerland: Springer, 2020, pp. 187–199, doi: 10.1007/978-3-030-45183-7_14.

[34] H. Amazal and M. Kissi, “A new big data feature selection approach for text classification,” *Sci. Program.*, vol. 2021, Art. ID 6645345, pp. 1–10, 2021.

[35] R. H. W. Pinheiro, G. D. C. Cavalcanti, and T. I. Ren, “Data-driven global-ranking local feature selection methods for text categorization,” *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1941–1949, Mar. 2015, doi: 10.1016/j.eswa.2014.10.011.

[36] R. C. P. Fragoso, R. H. W. Pinheiro, and G. D. C. Cavalcanti, “Data-driven feature selection methods for text classification: An empirical evaluation,” *J. Universal Comput. Sci.*, vol. 25, no. 4, pp. 334–360, 2019.

[37] D. Li, Z. Tan, and H. Liu, “Exploring large language models for feature selection: A data-centric perspective,” *SIGKDD Explor. Newsl.*, vol. 26, no. 2, pp. 44–53, Dec. 2024, doi: 10.1145/3715073.3715077.

[38] C.-F. Tsai and Y.-T. Sung, “Ensemble feature selection in high-dimension, low-sample-size datasets: Parallel and serial combination approaches,” *Knowl.-Based Syst.*, vol. 203, Art. no. 106097, Sep. 2020, doi: 10.1016/j.knosys.2020.106097.

[39] J. Arora, U. Agrawal, P. Tiwari, D. Gupta, and A. Khanna, “Ensemble feature selection method based on recently developed nature-inspired algorithms,” *in Proc. Int. Conf. Innovative Computing and Communications*, vol. 1087. Singapore: Springer, 2020, pp. 457–470.

[40] A. Khurana and O. P. Verma, “Novel approach with nature-inspired and ensemble techniques for optimal text classification,” Multimedia Tools Appl., vol. 79, pp. 23821–23848, 2020, doi: 10.1007/s11042-020-09013.

[41] S. Sharma and A. Jain, “Hybrid ensemble learning with feature selection for sentiment classification in social media,” *Int. J. Inf. Retr. Res.*, vol. 10, no. 2, p. 19, 2020.

[42] D. García-Gil et al., "Smart data based ensemble for imbalanced big data classification," *arXiv:2001.05759*, Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.05759

[43] R.-J. Palma-Mendoza, L. de-Marcos, D. Rodriguez, and A. Alonso-Betanzos, "Distributed correlation-based feature selection in Spark," *Inf. Sci.*, vol. 496, pp. 287–299, 2019, doi: 10.1016/j.ins.2018.10.052.

[44] A. Khodaei et al., "PRCFX-DT: A new graph-based approach for feature selection and classification of genomic sequences," *BMC Bioinformatics*, vol. 26, p. 159, 2025, doi: 10.1186/s12859-025-06183-4.

[45] R. C. P. Fragoso, R. H. W. Pinheiro, and G. D. C. Cavalcanti, "A method for automatic determination of the feature vector size for text categorization," *in Proc. 5th Brazilian Conf. Intelligent Systems (BRACIS)*, Recife, Brazil, Oct. 2016, pp. 259–264, doi: 10.1109/BRACIS.2016.055.

[46] A. Ben Brahim and M. Limam, "Ensemble feature selection for high-dimensional data: A new method and a comparative study," *Adv. Data Anal. Classif.*, vol. 12, no. 4, pp. 937–952, Dec. 2018, doi: 10.1007/s11634-017-0285-y.

[47] B. Pes, "Ensemble feature selection for high-dimensional data: A stability analysis across multiple domains," *Neural Comput. Appl.*, vol. 32, pp. 5951–5973, 2020, doi: 10.1007/s00521-019-04082-3.

[48] B. Pes, N. Dessì, and M. Angioni, "Exploiting the ensemble paradigm for stable feature selection: A case study on high-dimensional genomic data," *Inf. Fusion*, vol. 35, pp. 132–147, May 2017, doi: 10.1016/j.inffus.2016.10.001.

[49] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," *in Proc. 21st Int. Conf. Mach. Learn. (ICML)*, Banff, AB, Canada, 2004, p. 18, doi: 10.1145/1015330.1015432.

[50] S. Salloum, J. Z. Huang, and Y. He, "Random sample partition: A distributed data model for big data analysis," *IEEE Trans. Ind. Informatics*, vol. 15, no. 11, pp. 5846–5854, Nov. 2019, doi: 10.1109/TII.2019.2912723.

[51] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.

[52] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 20, pp. 832–844, 1998.

[53] C. Zhang and Y. Ma, Eds., *Ensemble Machine Learning : Methods and Applications*. New York, NY, USA: Springer, 2012.

[54] M. Vojnovic, F. Xu, and J. Zhou, "Sampling based range partition methods for big data analytics," *Microsoft Tech. Rep. MSR-TR-2012-18*, Mar. 2012. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sampling-based-range-partition-methods-for-big-data-analytics/

[55] M. S. Mahmud et al., "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Min. Anal*., vol. 3, no. 2, pp. 85–101, Jun. 2020, doi: 10.26599/BDMA.2019.9020015.

[56] S. Salloum, J. Z. Huang, Y. He, and X. Chen, "An asymptotic ensemble learning framework for big data analysis," *IEEE Access*, vol. 7, pp. 3675–3693, 2019, doi: 10.1109/ACCESS.2018.2889355.

[57] T. Z. Emara and J. Z. Huang, "RRPlib: A Spark library for representing HDFS blocks as a set of random sample data blocks," *Sci. Comput. Program*., vol. 184, Art. no. 102301, Oct. 2019, doi: 10.1016/j.scico.2019.102301.

[i] https://spark.apache.org/
[ii] https://nltk.org