2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC) | 978-1-6654-3579-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/EDOC52215.2021.00015

Scalable Global Mutual Information Based Feature Selection Framework for Large Scale Datasets

1st Majid Soheili
Computer Engineering Department
Neka Branch, Islamic Azad University
Neka, Iran
soheili@iauneka.ac.ir

2nd Maryam Amir Haeri
Learning, Data-Analytics and Technology Department
University of Twente
Enschede, Netherlands
m.amirhaeri@utwente.nl

***Abstract*—In the Big Data era, scalability is an essential characteristic of machine learning algorithms. Most data discovery algorithms apply a feature selection (FS) method as a crucial preprocessing step. The main objective of FS is to select a subset of informative features in such a way that the discriminating power will be kept. Unluckily, most traditional feature selection algorithm is not scalable, which is a significant weakness in coping with big datasets. This paper proposes a distributed and Scalable Global Mutual Information-based feature selection framework called SGMI to deal with large-scale datasets.**

The framework first generates a similarity matrix to representing dependency among all features. To this aim, the joint values histograms of paired columns are generated in a scalable way and a single pass. Next, based on these histograms, the dependency criterion elements, including individual and joint entropies, are extracted independently. Finally, the SGMI framework applies an optimization method to make feature rankings based on the similarity matrix. In this paper, three popular optimization methods, Quadratic Programming (QP), Spectral Relaxation (SR), and Truncated Power (TP), are plugged into the proposed framework. Consequently, three scalable FS methods, called SGMI-QP, SGMI-SR, SGMI-TP, will be produced. The experimental studies are performed on four balanced and imbalanced large-scale datasets. Then, the empirical outcomes are compared with a distributed feature selection method, DiRelief, and the original version of the produced methods.

The experimental results illustrate that (i) all produced methods are scalable and have a lower execution time than their traditional version and DiRelief method. (ii) SGMI-QP has a lower execution time than the two others. (iii) There is no significant difference among produced methods outcomes on experimental balanced big datasets. (iv) Generally, SGMI-SR produces better results to cope with big datasets than SGMI-QP, SGMI-TP, and DiRelief.

***Index Terms*—Scalable Feature Selection, Distributed Feature Selection, Mutual Information, Large-Scale Dataset**

I. Introduction

Data is growing in the whole world at an unprecedented rate, which has been estimated that 2.5 exabytes of data are generated per day [1]. Such voluminous data, which has a wide variety, high complexity, and high speed generating rate, is known as Big Data. This data includes valuable knowledge and insights, and thus prestigious companies and organizations try to find new ways to acquire these insights. However, traditional data discovery algorithms have been designed to cope with small or medium datasets, loaded into a single machine, and computed in a centralized environment. Nevertheless, in the Big Data era, in which there is no interruption in increasing data size, these algorithms do not scale properly. Therefore distributed and scalable learning algorithms have become essential.

Feature selection (FS) is a dimensionality reduction technique that is known as an essential preprocessing technique applied in various applications such as machine learning, data mining, and pattern recognition [2]–[6]. This technique has two major objects: simultaneously selecting the more informative features and discarding redundant features [7]. These purposes would be more eligible when a large-scale dataset needs to be processed because reducing feature space size will shrink the computation cost of the data mining algorithm.

According to dependency on a classification model, FS algorithms can be categorized into three groups: wrapper methods [8], [9], embedded methods [10], and filter methods [11]. Filter methods only rest on data's statistical properties, and since they are independent of any learning model, they prevent incurring a high computational cost and provide more generality than two other categories [12]. These advantages cause most proposed FS algorithms to cope with large-scale datasets be related to the filter category [13].

From another standpoint, according to the final result, FS methods can be organized into two groups: feature subset selection (FSS) methods and feature ranking (FR) methods. FSS methods search for a subset of informative features that provide a proper predictive capability approximately equal to the original features set. Meanwhile, FR methods rank the features based on some criteria such that more informative features will be inserted at the top of the final ranking [14].

During the last decade, mutual information (MI) is a well-known measure for evaluating the dependency between attributes, and it has been utilized by FS algorithms vastly. These algorithms have increased attention over the years because they are efficient and easy-used methods, and their theoretical background returns to the information theory [15].

The state-of-the-art MI-based feature selection methods belong to the filter category, and they can be categorized into two subgroups, (i) Greedy Feature Selector (ii) Global Feature Selector [16]

978-1-6654-3579-6/21/$31.00 ©2021 IEEE
DOI 10.1109/EDOC52215.2021.00015

The greedy approach can lead to a suboptimal result, whereas it is conventional in published MI-based FS methods. In this approach, the selected features can not be removed in later stages. On the opposite, the global approach applies a global optimization method such that it causes to gain a better performance potentially by considering the interaction among all features concurrently.

Global MI-based FS methods need a similarity matrix representing non-linear dependencies among all features and dependencies among features and class labels. However, producing the similarity matrix is a slow step in these methods, especially when the number of instances dominates the number of features. Moreover, once a given dataset is distributed in data nodes upon a distributed file system, producing the similarity matrix can be more time-consuming because it is necessary to transfer intermediate results among network links, known as the communication cost. Consequently, computing the similarity matrix forms a significant portion of the FS method execution time, and it overwhelms the time of features ranking.

Apache Spark is a well-known data analysis and machine learning framework for processing large-scale datasets. This framework applies MapReduce (MR) [17] as a standard paradigm for distributed processing [18]. Also, it applies an efficient data abstraction named resilient distributed dataset (RDD) and provides a fast execution environment for memory-based algorithms, which are common in the data analysis scope. The researchers have paid attention to this framework in recent years [19]–[22]. The proposed framework is implemented based on the Apache Spark computing model, whereas there is no tight dependency.

In this paper, a Scalable Global Mutual Information-based feature selection framework, called SGMI henceforth, is proposed to cope with large-scale datasets in the distributed environment. To generate the similarity matrix, the SGMI computes joint values histograms of all paired columns in a distributed way. Next, based on these joint histograms, the required components of the similarity matrix, including entropies and mutual information, will be extracted. The advantages of this method are; first, the dependency between all paired columns is computed in various worker nodes independently; second, the similarity matrix is generated in a single pass. Finally, three optimization methods, Quadratic Programming (QP), Spectral Relaxation (SR), and Truncated Power (TP), are plugged in the framework, whereas each arbitrary optimization method for ranking features can be applied. Consequently, three algorithms, SGMI-QP, SGMI-SR, and SGMI-TP, are produced by relying on the proposed framework.

For assessing the SGMI framework's performance, three produced algorithms are executed over a real cluster of computers and on four large-scale and high-dimensional datasets, two balanced and two imbalanced. Then toward performing a more profound study, empirical outcomes are confronted with the traditional version of the produced algorithms and a distributed feature ranking algorithm DiRelief [20]. The main contributions with this work are as follows:

- Enriching feature ranking methods in large-scale machine learning libraries such MLIB, a well-known library based on Apache Spark. In this case, a distributed and scalable global MI-based feature ranking framework is proposed, which is currently under-explored.
- Redesigning the classical feature ranking methods to solve their scalability issue and generating the similarity matrix in a distributed way and a single pass, a computation bottleneck.
- Analyzing the time complexity and scalability parameters of the proposed framework.

The rest of this paper is structured as follows. A short overview of related works is presented in section II. The details of global MI-based feature selection methods are described in section III. In section IV, the main idea of the SGMI framework and three produced methods are explained extensively. The experimental study will be explained in section V. Finally, section VI exposes concluding notes and prominent issues for future works.

II. Related Works

In this section, some papers are investigated in which a feature selection algorithm was presented in a distributed approach, and experiments were performed to deal with large-scale datasets.

Peralta et al. [23] introduced a feature selection algorithm based on the genetic algorithm to cope with big data datasets in 2015. The proposed method utilizes binary-coded vectors as initial offsprings that represent selected features. At the first step, the proposed method executes the genetic algorithm in each data partition and generates the intermediate result independently. Then, the proposed method aggregates the intermediate results at the second step to produce the final weight vector by applying the arithmetic mean.

Eiras et al.performed a comparative study between approaches, multithreading through the Weka and parallelization by the Apache Spark, to cope with big datasets. In this study, four well-known feature selection algorithms are investigated, and the two measures, including the execution time and the speed-up, are considered. The authors concluded that the algorithms implemented based on the Apache Spark parallelization have significantly better execution times than those implemented based on the multithreading approach [24].

Ramirez et al.proposed an FS method called fast-mRMR, a distributed version of the traditional FS method mRMR in 2016. This method was implemented on the Apache Spark in dealing with large-scale datasets. Experiments were performed on three big datasets, and the speed-up measure was investigated. The results depicted that the proposed method has a proper speed-up [19].

Dagida et al.proposed two FS methods based on the rough set theory in two separate papers [25], [26]. The first method is a rough set theory-based FS method adapted to the Apache Spark computing model, and it is called dRST.In this method, the space of the original features is partitioned randomly. The second method, named LSH-dRST, applies the Locality

Sensitive Hashing (LSH) to put similar features into the same buckets. Then these buckets are mapped to data partitions to make possible appropriately splitting of the universe. According to experiments, the authors said that the second method is more reliable and scalable than the first.

Mendoza et al.introduced an FS method called DiReliefF in 2018 [20], and It is a distributed version of the traditional and well-known FS algorithm ReliefF.First, the experimental studies are carried out on four big datasets, and the memory consumption and execution time a re investigated. Then, a comparison between the proposed method and the classical version of ReliefF is performed. The empirical results illustrate that the proposed method has proper scalability and can handle large-scale datasets efficiently.

Mendoza et al.published their second work based on the classical Correlation-based Feature Selection (CFS) method, and they proposed the horizontally and vertically distributed versions of this traditional method labeled DiCFS-hp, DiCFS-vp [12]. For the experimental study, four big datasets utilized in their previous work are considered, and a comparison between proposed methods and non-distributed CFS was performed. The results depict that the proposed methods have better execution time and scalability in comparison with the classical version of CFS.

Ramirez et al.published their later paper in which an information-theory-based feature selection framework is proposed in confronting large-scale datasets [27]. According to the authors' claim, the proposed framework can be applied for most information-theory-based feature selection methods. In experiments, the distributed version of traditional mRMR confronts four large-scale datasets, and then measures including execution time and Area Under Curve (AUC) are considered. Finally, the authors concluded that their method could handle ultra-high dimensional large-scale datasets.

Soheili et al.proposed a scalable quadratic programming feature selection method called DQPFS [28], which requires seeking the whole dataset in three passes. The authors compared the introduced method and the non-distributed version of QPFS [29] as the empirical study. The experimental results depict that the proposed method has a lower execution time and comparable accuracy than its classical version.

To summarize, in the literature, some feature selection methods have been proposed for distributed environments; however, to the best of our knowledge, there is no research to proposed a general distributed and scalable framework for covering most global MI-based feature selection methods as proposed in this paper.

III. Global MI-based Feature Selection

Most FS methods, which apply the MI, consider two major concepts, $relevancy$, and $redundancy$, such that the former is useful and the latter is a harmful factor. An FS algorithm tries to find a subset of features in such a manner that the selected features have the least redundancy among themselves and the most relevancy with the class label at the same time. Given a dataset with a set of features $\mathbb{X} = \{X_1, \ldots, X_d\}$, and a class label C, the FS algorithm maximizes equation (1), and as the final result, $\mathbb{S}$ is a set of selected features.

$$\max_{X_i \in \mathbb{X} \setminus \mathbb{S}} \{Rel\left(X_i\right) - Red\left(X_i|\mathbb{S}\right)\} \tag{1}$$

The Minimum Redundancy Maximum Relevance (MRMR) framework [30] is a well-known FS method that defines $relevancy$ and $redundancy$ based on MI as equations (2) and (3), respectively. These definitions reveal that the MRMR framework utilizes a greedy approach for selecting an individual feature in each iteration in such a manner that the objective in (1) be maximized. Therefore, this method can be lead to a suboptimal result, due to a feature selected in an earlier stage cannot be removed at a later stage [15]. Note that, in equations (2) and (3), the $I\left(\cdot\right)$ refers to the MI.

$$Rel(X_i) \triangleq I\left(X_i; C\right) \tag{2}$$

$$Red(X_i|\mathbb{S}) \triangleq \frac{1}{|\mathbb{S}|} \sum_{X_j \in \mathbb{S}} I\left(X_i; X_j\right) \tag{3}$$

As mentioned before, the MRMR applies an incremental approach, but its idea can be represented as a general global feature subset selection problem as (4). It is noticeable that in (4), the k refers to the number of selected features, and also, the α is a weight factor used to balance two components $relevancy$ and $redundancy$.

$$\text{FSS}_{MI} : \max_{\substack{\mathbb{S} \subset \mathbb{X} \\ |\mathbb{S}|=k}} \left\{ \sum_{X_i \in \mathbb{S}} I\left(X_i; C\right) - \alpha \sum_{\substack{X_i, X_j \in \mathbb{S} \\ i \neq j}} I\left(X_i; Xj\right) \right\} \tag{4}$$

As the first attempt to solve this problem, the Quadratic Program Feature Selection (QPFS) [18] algorithm is proposed. This method reformulates equation (4) as quadratic programming with linear constraints, which is presented in (5). In (5), Q is a symmetric positive semidefinite matrix, and F is a d-dimensional vector with non-negative entries. After solving (5), the weight of features is determined as another d-dimensional vector w. The QPFS is convex quadratic programming, and there are polynomial time published algorithms to provide an optimal global solution. The matrix Q represents the redundancy among features such that $Q_{ii} = H\left(X_i\right)$, $Q_{ij} = I\left(X_i; X_j\right),\ i \neq j$, and $H\left(\cdot\right)$ refers to the entropy function. Also, the vector F represents the relevancy among features and class label such that $F_i = I\left(X_i; C\right)$.

$$\text{QP: } \min_{w} \left\{\alpha w^T Q w - w^T F\right\} \ \ s.t. \sum_{i=1}^{d} w_i = 1, w_i \geq 0 \tag{5}$$

Moreover, the global feature subset selection was proposed as another equation (6) based on Conditional Mutual Information (CMI) [15]. This equation is redefined in the form of a quadratic integer programming problem as equation (7). As an advantage, this equation needs no improvised parameters,

including the balancing and convexification parameters. However, as a disadvantage, this equation is more complex and has more computation cost than the previous version, and applying a more complex similarity matrix will not lead to better outcomes necessarily. This fact is illustrated in Section V-B1.

$$\text{FSS}_{CMI} : \max_{\substack{\mathbb{S}\subset\mathbb{X} \\ |\mathbb{S}|=k}} \left\{ \sum_{X_i\in\mathbb{S}} I\left(X_i;C\right) + \sum_{X_i,X_j\in\mathbb{S}} I\left(X_i;C|X_j\right) \right\} \quad (6)$$

$$\text{QIP}_{CMI} : \max_{w} \left\{ w^T Q w \right\} s.t.\ w \in \{0,1\}^d, \|w\| = \sqrt{k} \quad (7)$$

In equation (7), the Q is a hessian matrix that integrates both the concept of relevancy and total redundancy such that $Q_{ii} = I\left(X_i;C\right)$ and $Q_{ij} = \frac{1}{2}\left\{I\left(X_i;C|X_j\right) + I\left(X_j;C|X_i\right)\right\}, i \neq j$. In literature, for solving equation (7), some methods are proposed, such as Spectral Relaxation (SRFS), Semidefinite Programming (SDFS), Truncated Power Method (TPFS), and Low-Rank Bilinear Approximation (LRFS). Nevertheless, profoundly investigating these methods is beyond this paper's objective, and their details can be found in [15], [16].

Generally, the complexity of these global MI-based feature selection methods includes two components, the complexity of the making similarity matrix and the complexity of features ranking. The similarity matrix complexity in traditional methods is $O\left(nd^2\right)$, in which n refers to the number of instances and d refers to the number of columns. Also, their ranking features complexities are different and noted in the third column of Table I. Note that when the number of instances is so larger than the number of features $(n \gg d)$, the total complexity of these traditional methods is equal to the similarity matrix complexity. The summarization of these methods and their time complexities are presented in Table I.

As mentioned before, in confronting a large-scale dataset, generating the similarity matrix takes a large portion of the total execution time. Hence, in this paper, a global MI-based feature selection framework is proposed that generates the similarity matrix in a scalable way. Moreover, It can utilize each method listed in Table I for ranking features as a global optimization method.

IV. SGMI Framework

In this section, the proposed framework is explained comprehensively. As mentioned before, the SGMI is a distributed and scalable feature selection framework such that it tries to find a feature ranking as a global optimum solution upon an MI-based similarity matrix. However, the execution time of generating the similarity matrix overwhelmed the total execution time, particularly once the dataset is distributed in several data nodes. In this situation, calculating dependency among features needs to transfer data among network links, which is significantly time-consuming. Therefore, generating the similarity matrix is a bottleneck to confront a large-scale dataset. Thus, the matrix must be computed in a distributed and scalable approach.

A. Similarity Matrix in Theory

As mentioned in section III, the similarity matrix could be generated based on the MI or CMI, defined by (8), and (9), such that $H\left(\cdot\right)$ refers to entropy. The entropy of a given histogram of values $\mathbb{F} = \{f_0, \ldots, f_b\}$ is computed as (10) such that $n = \sum_{i=0}^{b} f_i$. It worth mentioning that b refers to the number of unique values in a nominal feature. In other words, b refers to the number of bins applied to discretizing continuous features. According to equations (8) to (10), generating the similarity matrix could be reduced to generating histograms of values.

$$I\left(X_i;X_j\right) = H\left(X_i\right) + H\left(X_j\right) - H\left(X_i,X_j\right) \quad (8)$$

$$\begin{aligned} I\left(X_i;C|X_j\right) = {} & H\left(X_i,X_j\right) + H\left(X_j,C\right) \\ & - H\left(X_i,X_j,C\right) - H\left(X_j\right) \end{aligned} \quad (9)$$

$$H\left(\mathbb{F}\right) = -\sum_{i=0}^{b} \left(\frac{f_i}{n} \times \log \frac{f_i}{n} \right) \ \ s.t. \ \ f_i \neq 0 \quad (10)$$

The noticeable point is that computing the individual entropy and joint entropy would be feasible by generating the joint value histogram vector. Therefore, if the joint value histogram of two features X_i and X_j is represented as a square matrix $\mathbb{F} \in \mathbb{N}^{b\times b}$ such that $f_{i,j}$ is equal to the frequency of joint values (x_i, x_j), the joint and individual entropies will be computed based on equations (11) to (13). Therefore, acquiring the joint value histogram of two features, the MI measure between them can be computed in a single pass.

$$\begin{aligned} & H\left(X_i;X_j\right) \triangleq H\left(\mathbf{F}_{xy}\right) \\ & \quad s.t.\ \mathbf{F}_{xy} = \{f_{i,j} \in \mathbb{F} \mid \forall i,j \in \{0,\ldots,b\}\} \end{aligned} \quad (11)$$

$$H\left(X_i\right) \triangleq H\left(\mathbf{F}_x\right) \ \ s.t.\ \mathbf{F}_x = \left\{ \sum_{j=0}^{b} f_{i,j} \in \mathbb{F} \mid \forall i \in \{0,\ldots,b\} \right\} \quad (12)$$

$$\begin{aligned} & H\left(X_j\right) \triangleq H\left(\mathbf{F}_y\right) \\ & s.t. \mathbf{F}_y = \left\{ \sum_{i=0}^{b} f_{i,j} \in \mathbb{F} \mid \forall j \in \{0,\ldots,b\} \right\} \end{aligned} \quad (13)$$

Assuming that the joint value histogram of three arbitrary columns $X_i,\ X_j, C$ represented as a cube $\mathbb{F} \in \mathbb{N}^{b\times b\times b}$ such that $f_{i,j,k}$ is equal to the frequency of joint values (x_i, x_j, c_k), the joint and individual entropies can be computed as equations (14) to (17). Therefore, acquiring the joint value histogram of three columns, the CMI measure can be computed in a single pass.

$$\begin{aligned} & H\left(X_i;X_j;C\right) \triangleq H\left(\mathbf{F}_{xyz}\right) \\ & \quad s.t.\ \mathbf{F}_{xyz} = \{f_{i,j,k} \in \mathbb{F} \mid \forall i,j,k \in \{0,\ldots b\}\} \end{aligned} \quad (14)$$

$$\begin{aligned} & H\left(X_i,X_j\right) \triangleq H\left(\mathbf{F}_{xy}\right) \\ & \quad s.t.\ \mathbf{F}_{xy} = \left\{ \sum_{k=0}^{b} f_{i,j,k} \in \mathbb{F} \mid \forall i,j \in \{0\ldots b\} \right\} \end{aligned} \quad (15)$$

TABLE I: The summarization of ranker methods applied to solve the global feature selection problem.

Global FSS	Method	Complexity		Similarity Definition
		Ranking	Similarity	
FSS_{MI}	QPFS	$O\left(d^3\right)$	$O\left(nd^2\right)$	$Q_{ii}=H(X_i)$ $Q_{ij}=I(X_i;X_j),\ i\neq j$
FSS_{CMI}	SRFS	$O\left(d^2\right)$	$O\left(nd^2\right)$	$Q_{ii}=I(X_i;C)$ $Q_{ij}=\frac{1}{2}\{I(X_i;C\|X_j)+I(X_j;C\|X_i)\},i\neq j$
	SDFS	$O\left(d^{4.5}\right)$		
	TPFS	$O\left(td^2\right)$		
	LRFS	$O\left(n^{(k+1)}\right)$		

t is number of iteration, k is number of approximate rank

$$H\left(X_j, C\right) \triangleq H\left(\mathbf{F}_{yz}\right)$$
$$s.t.\ \mathbf{F}_{yz} = \left\{ \sum_{i=0}^{b} f_{i,j,k} \in \mathbb{F} \mid \forall j, k \in \{0 \dots b\} \right\} \tag{16}$$

$$H\left(X_j\right) \triangleq H\left(\mathbf{F}_{y}\right)$$
$$s.t.\ \mathbf{F}_{y} = \left\{ \sum_{i=0}^{b} \sum_{k=0}^{b} f_{i,j,k} \in \mathbb{F} \mid \forall j \in \{0, \dots b\} \right\} \tag{17}$$

According to this section's equations, generating the similarity matrix will be feasible by generating the required joint histograms. This solution has two prominent advantages firstly, generating joint histograms in a single pass instead of exploring data partitions multiple times; secondly, computing the required individual and joint entropies in various worker nodes independently instead of transferring computation cost to the master node. Note, the joint value histogram vectors include lots of zero values because most combinations of joint values do not occur, especially when the three columns are considered; hence histograms are presented as sparse vectors.

B. Steps of SGMI Framework

As mentioned before, generating the similarity matrix is so time-consuming that it is a computing bottleneck in a Global MI-based FS method. The SGMI framework makes the similarity matrix in a distributed and scalable way, and then one of the optimization methods mentioned in Table I can be utilized to ranking the features. The main procedure pseudocode of the proposed framework implemented based on the Apache Spark Computing Model is detailed by Algorithm 1.

In Algorithm 1, the DS refers to the given dataset with n instances and d columns such that the class label is placed in the last column. Moreover, the proposed framework utilizes information theory. Hence continuous features are discretized into certain bin buckets before entrancing the framework. The number of bins is determined as the input parameter b. Input parameter $\mathbf{M}$ refers to a feature ranking method listed in Table I, and it is applied to the $similarity$ matrix. As the final result, the output parameter $\mathbf{R}$ refers to a ranking of features.

As Fig. 1 depicts graphically, the SGMI framework has five steps, and in the following, the details of these steps will be explained extensively. Note that, Algorithm 1 and Fig. 1 describe generating similarity matrix based on the CMI.

The First Step, Transforming to Columnar Format: To generate the similarity matrix, it needs to compute the dependencies among features; hence each feature is fetched

Algorithm 1: SGMI- Main Procedure

Input : $DS \in \mathbb{N}^{n \times d}$ // the given dataset with d columns
Input : $b \in \mathbb{N}$ // the maximum number of bins
Input : $\mathbf{M}$ // a method mentioned in TableI
Output: $\mathbf{R}$ // the feature ranking
$DS_c = \text{TCF}\left(DS\right)$ // applying Algorithm 2
$histogram = \text{DH}\left(DS_c, b\right)$ // applying Algorithm 3
$entropies = \text{DE}\left(histogram\right)$ // applying Algorithm 5
$nf = d - 1$ // the number of features
$similarity = \text{new Matrix}\left[nf, nf\right]$
for $i = 0$ **to** nf **do**
  for $j = i$ **to** nf **do**
    if $i == j$ **then**
      $similarity\left[i, j\right] = I\left(X_i; X_d\right)$ // applying (8)
    else
      $similarity\left[i, j\right] = \frac{1}{2}\left\{I\left(X_i; X_d | X_j\right) + I\left(X_j; X_d | X_i\right)\right\}$ // applying (9)
      $similarity\left[j, i\right] = similarity\left[i, j\right]$
    end
  end
end
$weights = \mathbf{M}\left(similarity\right)$ // applying a ranker method
$\mathbf{R} = \text{Order}\left(weights, descending = T\right)$
return $\mathbf{R}$

multiple times. To decreasing features access time during generate similarity matrix, each data partition is transformed into a columnar format such that each column is converted to a row. Therefore, only reading a specific row is needed for reading a feature in the new columnar format instead of seeking a whole dataset. This idea was applied in [27] formerly, and Algorithm 2 explains it. The output of Algorithm 2 is the transformed dataset whose row number is equal to $n' = p \times d$ and whose column number is equal to $d' = \frac{n}{p}$ on average. Note that variables n, p, and d respectively refer to the number of instances, columns, and data-partitions of the given dataset, respectively.

The Second Step, Joint Value Histogram: As mentioned before, in section IV-A, the joint value histogram is needed for computing the dependencies based on CMI among three arbitrary features. To this aim, each data partition is transformed into collections of $\langle key, value \rangle$ such that the key refers to indexes of features, and the $value$ refers to the histogram vector. After completing the first transformation of this step, each data partition includes the partial histogram vectors. Therefore, to make full histogram vectors, partial histogram vectors with the same keys must be shuffled and merged. To

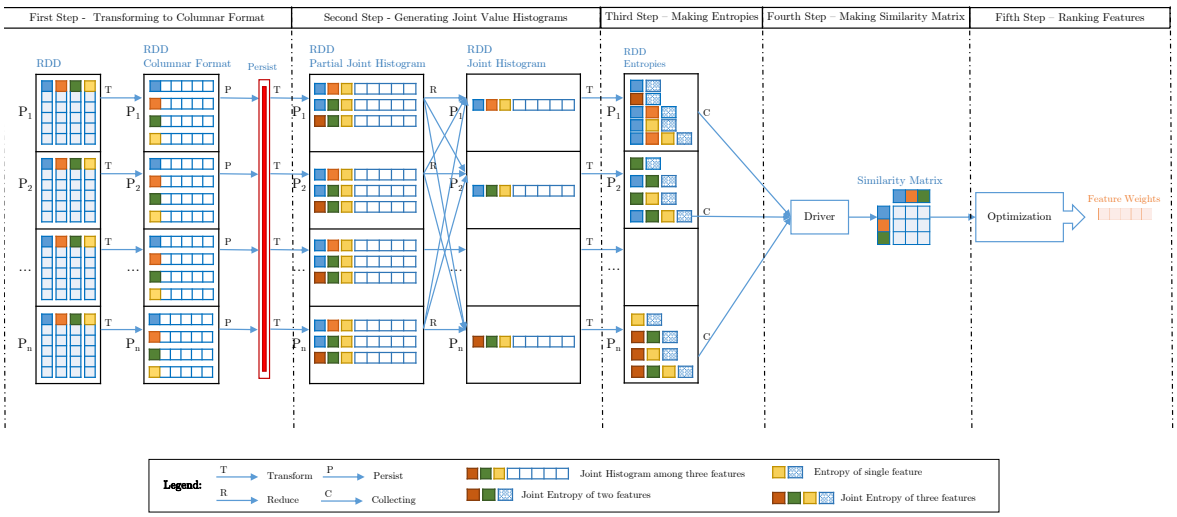


Fig. 1: The scheme of the SGMI framework

Algorithm 2: TCF- Transform to the Columnar Format

Input : $DS \in \mathbb{N}^{n \times d}$ // the given dataset
Output: $DS_c \in \mathbb{N}^{n' \times d'}$ // the columnar format of DS

// n' will be equal to $p \times d$
// d' will be equal to $\frac{n}{p}$ on average
// p is equal to the number of partition of DS

1 $DS_c =$ **forall** $part \in DS$ **do in parallel**
2 $d' = part.length$ // the number of instances per partition
3 $part_c =$ new Matrix $[d, d']$
4 **for** $i = 1$ **to** d' **do**
5 **for** $j = 1$ **to** d **do**
6 $part_c[j, i] = part[i, j]$
7 **end**
8 **end**
9 $part_c$
10 **end**
11 **return** $DS_c.persist$

generate the similarity matrix based on the MI, computing the full histogram vectors of two columns is needed, whereas generating the similarity matrix based on the CMI needs to compute three columns' full histogram vectors. It worth mentioning that the second step of Fig. 1 shows the histogram vectors for computing the CMI. Algorithm 3 details this step. In Algorithm 3, DS_c refers to the given dataset in the columnar format. The input parameter b refers to the maximum number of bins in features, and it is passed to Algorithm 4. The output of Algorithm 3 is a distributed collection of joint value histograms.

The Third Step, Making Entropies: At the end of the previous step, all data partitions are transformed into various joint value histograms. In this step, according to the equations (14) to (17) are mentioned before, sub histograms extract from each original histogram, then they are transformed into the corresponding entropies. Note that computed entropies are presented as a dictionary such that each entropy can be access

Algorithm 3: DH- Distributed Histogram

Input : $DS_c \in \mathbb{N}^{n' \times d'}$ // the given dataset in columnar format
Input : $b \in \mathbb{N}$ // the maximum number of bins
Output: $histogram \in \mathbb{N}^{d^2 \times b^3}$

// d will be equal $\frac{n'}{p}$

1 $partial =$
2 **forall** $part_c \in DS_c$ **do in parallel**
3 $d = part_c.length$ // the number of features
4 $hist =$
5 **for** $i = 0$ **to** $d - 1$ **do**
6 **for** $j = i$ **to** $d - 1$ **do**
7 $x = part_c[i, :]$
8 $y = part_c[j, :]$
9 $z = part_c[d, :]$
10 $\langle (i, j, k), \text{Histogram}(x, y, z, b) \rangle$ // applying Algorithm 4
11 **end**
12 **end**
13 **end**
14 $histogram =$
$partial$.reduceByKey {AggregateSparseVectors} // Full Histogram
15 **return** $histogram$

by a triple key. Algorithm 5 details this step.

The Fourth Step, Making Similarity Matrix: As mentioned before, the similarity matrix is symmetric, and it can be produced based on MI or CMI according to Table I. Each cell of the matrix is computed by utilizing entropies calculated in the preceding step and gathered in the master node. Noteworthy that, depending on the applied measurement, the similarity matrix cells are computed based on equations (14) to (17). Lines 5 to 15 of Algorithm 1 explain how the similarity matrix produce based on the CMI.

The Fifth Step, Ranking Feature: At the end of the fourth step, the similarity matrix is placed on the driver node. For ranking features, each optimization method mentioned in

Algorithm 4: Histogram

Input : $vec_1 \in \mathbb{N}^{d'}$ // the first vector
Input : $vec_2 \in \mathbb{N}^{d'}$ // the second vector
Input : $vec_3 \in \mathbb{N}^{d'}$ // the third vector
Input : $b \in \mathbb{N}$ // the maximum number of bins
Output: sv // a sparse vector of the joint values

$frequency =$ new Array$[b, b, b]$
for $i = 0$ **to** d' **do**
  $v_1 = vec_1\,[i]$
  $v_2 = vec_2\,[i]$
  $v_3 = vec_3\,[i]$
  $frequency\,[v_1, v_2, v_3]$ +=1
end
$sv =$ new SparseVector$(frequency)$
return sv

section IV-A can be applied. Indeed the optimization method executes in the driver node without any dependency on other data nodes. The outcome of the optimization method is a weight vector such that the most informative feature has the biggest value. Hence the feature weights are ordered descendingly.

Algorithm 5: DE - Distributed Entropies

Input : $histogram \in \mathbb{N}^{d^2 \times b^3}$
Output: $entropies \in$ Dictionary$[indexes, value]$ // dictionary of entropies

forall $(idxs, vec) \in histogram$ **do in parallel**
  $(i, j, k) = idxs$
  $C = X_k$
  $entropies\,[(i, j, k)] = H\,(X_i, X_j, C)$ // applying (14)
  $entropies\,[(i, j, nill)] = H\,(X_i, X_j)$ // applying (15)
  $entropies\,[(j, k, nill)] = H\,(X_j, C)$ // applying (16)
  $entropies\,[(j, nill, nill)] = H\,(X_j)$ // applying (17)
end
return $dictionary$.collect()

C. Time Complexity Analysis

As mentioned before, the complexity of the traditional global MI-based feature selection has two components: the complexity of the similarity matrix computing and the complexity of features ranking. In confronting a big dataset, the data has been split into different data partitions and distributed in several data nodes. The proposed framework applies a distributed approach to making the similarity matrix, but its feature ranking method is run on a driver/master node locally. The standard models of distributed computing typically assume that local computation is free, and it is rooted in an assumption that communication cost and data transferring among network links dominate the execution time of the local computations. Thus in this section, the complexity of the similarity matrix generated in a distributed approach is investigated. In other words, the complexity of the SGMI framework is a combination of the complexity of three algorithms 2, 3, and 5.

Assuming the instance number of the given dataset is depicted as n, and the number of data partitions is depicted as p, thus each data partition includes $\frac{n}{p}$ instances. Also, assuming the number of worker nodes is depicted as w, the number of data-partitions are processed by a worker node will be equal to $\lceil \frac{p}{w} \rceil$, on average.

By assuming the column number of the given dataset is depicted as d, the time complexity of Algorithm 2 is equal to (18). The noticeable point is that the output of this algorithm is a distributed dataset whose the numbers of rows and columns are equal to $n' = p \times d$, $d' = \frac{n}{p}$, respectively. Moreover, the time complexity of Algorithm 3 is equal to (19), when it receives the output of Algorithm 2 as an input parameter, and it can be reduced to (20). Also, the time complexity of Algorithm 5 is equal to (21). Note that the factor b is mentioned in section IV-B, and its value is much smaller than n, usually.

As a summarized the complexity of the SGMI framework is equal to (22). According to (22), the execution time of the SGMI framework in increased by growing the size of datasets, and also it decreased by increasing the number of worker nodes.

$$T_{tcf}\,(n, d) = \mathcal{O}\left(\frac{nd}{w}\right) \tag{18}$$

$$T_{dh}\,(n', d', b) = \max\left(\mathcal{O}\left(\frac{nd^2}{w}\right), \mathcal{O}\left(\frac{d^2 b^3}{w}\right)\right) \quad s.t \quad n' = p \times d, \quad d' = \frac{n}{p} \tag{19}$$

$$\text{if} \quad n > b^3 \Rightarrow \quad T_{dh}\,(n', d', b) = \mathcal{O}\left(\frac{nd^2}{w}\right) \tag{20}$$

$$T_{de}\,(d, b) = \mathcal{O}\left(\frac{d^2 b^3}{w}\right) \tag{21}$$

$$\text{if} \quad n > b^3 \Rightarrow$$
$$T_{sgmi}\,(n, d, b) = \max\left(\mathcal{O}\left(\frac{nd}{w}\right), \mathcal{O}\left(\frac{nd^2}{w}\right), \mathcal{O}\left(\frac{d^2 b^3}{w}\right)\right) = \mathcal{O}\left(\frac{nd^2}{w}\right) \tag{22}$$

V. Experimental Study

The produced methods are executed on four large-scale datasets for performing experimental studies, and then their final results are gathered. Next, the results are applied to reduce feature space, and then various classification Gaussian Naive Bayes and Random Forest models are induced based on the new feature space. Finally, the outcomes of the classifier models are considered as the performance of the produced methods. It is noteworthy that, for fitting classifier models to the training dataset, the 5-fold cross-validation method is applied. Moreover, the FS methods which applied information theory need to discretize continuous features. To this aim, all continuous features are quantized into ten bins such that the width of bins is equal.

The performance of the produced methods is compared with their traditional versions [15], [16], [18], and the DiRelief

TABLE II: The properties of the experimental datasets

#	Dataset name	Instances	Features	Minor:Major	Format	Class
1	Alpha	500,000	500	(50:50)	ASCII	2
2	OCR (sampled)	1,750,000	1156	(49:51)	Binary	2
3	FD (sampled)	2,000,000	900	(90:10)	Binary	2
4	ECBDL (sampled)	2,000,000	631	(98:2)	ASCII	2

algorithm, which is published recently [20]. The DiRelief algorithm is a distributed version of the popular ReliefF method, and Mendoza published the source code in the Repository[1]. The DiRelief algorithm requires two ad-hoc parameters, the number of samples and neighbors, which both are appointed to 20 in experiments. Note that feature ranking results of the produced methods are equal to their traditional version; therefore, the outcomes of classification models are equal as well, whereas their execution-time and their speed-up will be different from their traditional versions. For reproducibility objectives, the SGMI framework code has been uploaded to a repository[2].

A. Datasets and Computer Cluster

Four popular large-scale datasets are utilized in experiments that two first are balanced, and the rest are imbalanced. All datasets have a high number of features and a high number of instances such that the number of instances dominated the number of features. The experimental datasets' properties are presented in Table II. The datasets are accessible in the Pascal[3] and BDCOMP[4] repositories.

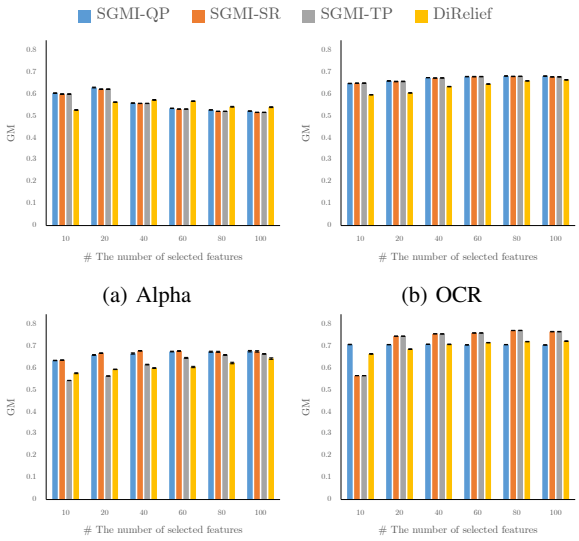
The experimental study is performed upon a real computer cluster with nine computing nodes. All nodes in the cluster have two-core processors, 8 gigabytes RAM, and the Apache Hadoop 2.7 and the Apache Spark 2.3 are installed on them.

B. Empirical Study

In a classification task, accuracy is a conventional criterion for evaluating a prediction model. Nevertheless, it is not proper to confront imbalanced datasets because minority classes have negligible effects on overall accuracy. Consequently, the weak performance of the minority class is dominated by the performances of the majority classes. Therefore the Geometric Mean (GM) Measurement is applied, and its detail is presented in the next section. Further to the GM measure, the Execution Time and The Speed-Up are considered in the experimental study. In the following, the empirical results based on the mentioned criteria are reported.

1) GM: The GM measurement aims to augment the accuracy of the minority and majority classes simultaneously. This measurement is presented based on two components, $sensitivity$ and $specificity$, such that they are computed based on the confusion matrix. These measures are computed as: $GM = \sqrt{sensitivity \times specificity}$, $sensitivity = \frac{TP}{TP+FN}$, and $specificity = \frac{TN}{FP+TN}$. In these equations, TP, TN, FP, and FN are True Positive, True Negative, False Positive, and False Negative, respectively.

[1]https://github.com/rauljosepalma/DiReliefF.
[2]https://github.com/Majid-Soheili/SGMI
[3]ftp://largescale.ml.tu-berlin.de/largescale/
[4]http://cruncher.ncl.ac.uk/bdcomp/



(a) Alpha (b) OCR

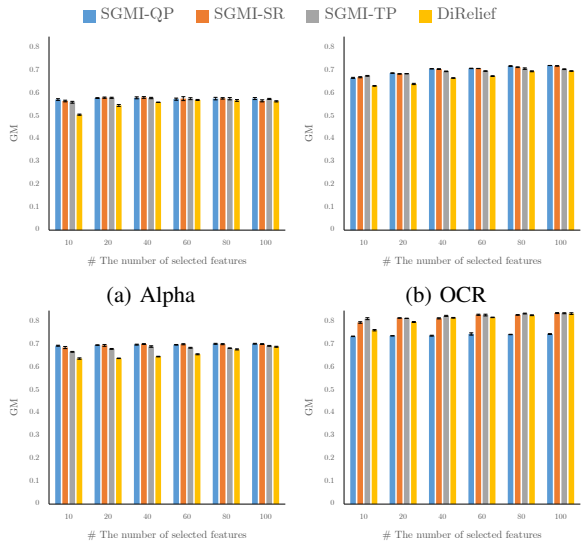
(c) ECBDL (d) FD

Fig. 2: The GM measurement results for balanced and imbalanced datasets by applying the Naive Bayes classifier

Figures 2a, 2b, 3a, and 3b depict that all produced methods have similar results by applying the different top selected features number to confront experimental balanced and big datasets based on the Naive Bayes and the Random Forest classifiers. Furthermore, Figures 2c, 2d, 3c, and 3d illustrate that in dealing with big imbalanced datasets, the SGMI-SR method has better or comparable results than SGMI-QP, and SGMI-TP, $20/24$ of experimental.It worth mentioning that the feature ranking results of produced methods are equal to their classical version outcomes. As an instance, the outcomes of the produced methods SGMI-QP and traditional QPFS are thoroughly the same. Thus in Fig. 2 and Fig. 3, there is no comparison among outcomes of the produced methods and classical versions. Also, compared with DiRelief, the produced algorithms have almost better results than DiReliefF.In detail, the SGMI-QP, SGMI-SR, and SGMI-TP have better results in 36, 43, and 41 of 48 experimental, respectively.

As a result, Fig. 2 and Fig. 3 illustrate that applying a more complex similarity matrix could not assure better outcomes, especially to cope with balanced and big datasets, whereas this matter could be more than justified in coping with imbalanced and big datasets. This matter is mentioned before in Section III.

2) Execution Time: The execution time of a distributed algorithm is a critical characteristic to deal with large-scale datasets. As mentioned before, owing to needing data transfer among worker nodes and under network links, calculating the similarity matrix takes a big portion of total execution time, such that it overwhelmed the execution time of the feature ranking method. In addition, the complexity of the dependency criterion applied in the similarity matrix is another effective factor in the execution time. Hence, the execution time of the produced method SGMI-SR is longer than SGMI-QP



(a) Alpha (b) OCR

(c) ECBDL (d) FD

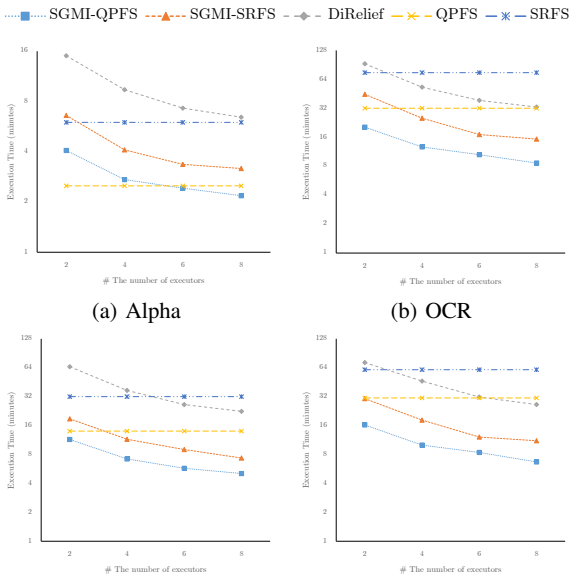
Fig. 3: The GM measurement results for balanced and imbalanced datasets by applying the Random Forest classifier.

because it utilizes a complex and time-consuming relation to compute the dependency among features. Also, the execution times of SGMI-SR and SGMI-TP methods are almost equal because both of them applied a similar relation to represent the similarity matrix. Hence in Fig. 4, SGMI-TP is not depicted because it would be placed behind the SGMI-SR line. Moreover, Fig. 4 shows that the produced methods' execution time is significantly shorter than the execution of their classical version in a single node with a single-core processor.

Another noticeable point is that the execution times of the SGMI-QP and SGMI-SR are lower than the DiRelief in all experiments as well. Note, some overheads such as the lineage graph can cause increasing produced algorithms' execution time, especially once the participating worker nodes are few.

3) Speed-Up: In the speed-up study, the number of participating executors increases while keeping the dataset size the same. In the ideal case, the execution time should be decreased linearly with increasing the number of executors. The speed-up measure can reveal how the proposed algorithms' execution time will be decreased by adding the computing resource. The speed-up would be calculated as $Speed\text{-}Up = \frac{\mathcal{T}(n,1)}{\mathcal{T}(n,w)}$. Notice that in the equation, $\mathcal{T}(n,w)$ represents the algorithm's execution time such that n and w relate to the instance number in the dataset and the number of participating worker nodes, respectively.

As Fig. 5 illustrates, produced methods have proper speed-up in all experimental datasets. However, the SGMI-SR has a comparable or a little better speed-up than SGMI-QP.Their speed-up decreases by increasing the number of participating worker nodes because by raising the number of worker nodes, communication costs will increase. As another point of view, the speed-up of the SGMI-SR is similar to the DiRelief algorithm, and both of them have more execution time than

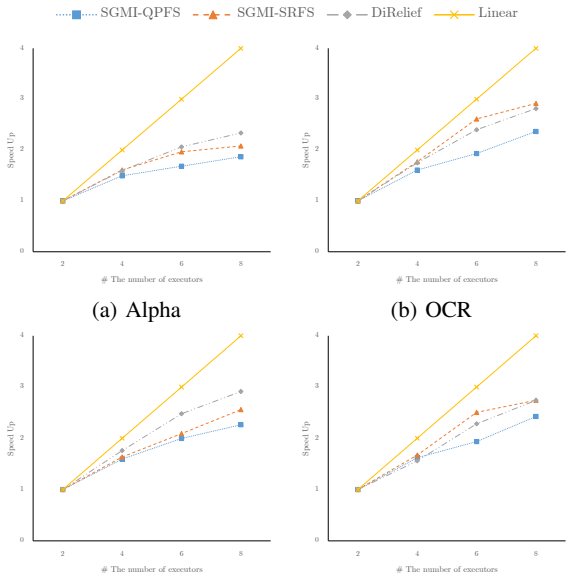


(a) Alpha (b) OCR

(c) ECBDL (d) FD

Fig. 4: A comparison among the execution time of the produced methods and their classical versions.

SGMI-QP, and it illustrates that when the computation cost is high, increasing the number of worker nodes can be more effective.



(a) Alpha (b) OCR

(c) ECBDL (d) FD

Fig. 5: The speed-up of the produced methods.

VI. Conclusion

In this paper, a distributed and scalable global MI-based feature selection framework, SGMI, is proposed. It firstly generates the similarity matrix in a scalable way and a single

pass. Next, it applies a feature ranking method to find a globally optimal solution upon the similarity matrix.

The similarity matrix represents the dependency among features simultaneously, and it can be computed based on the MI or CMI such that the first one has less complexity than the second one. In this paper, three optimization methods are applied in the SGMI framework such that the first one utilizes a MI similarity matrix and the rest of them use a CMI similarity matrix. In this circumstance, three methods, SGMI-QP, SGMI-SR, and SGMI-TP, are produced. As outcomes, these methods generate a feature ranking with the intent that informative features are placed on the top of the ranking.

The experimental studies are carried out on four big datasets, two balanced and two imbalanced, upon a computer cluster. Empirical results depict that generating the similarity matrix is so time-consuming that it dominates the feature ranking method's execution time. Hence, a similarity matrix that uses a more straightforward relation will have a lower execution time. Furthermore, the empirical results depict that SGMI-SR and SGMI-TP have similar execution times because using an equal similarity matrix. Moreover, the results illustrate no significant difference among the performance of produced methods to cope with balanced and big datasets, whereas in confronting the imbalanced and big dataset, SGMI-SR can potentially produce better results.

Compared with the DiRelief algorithm, produced algorithms have lower execution times and almost better GM results in confronting all experimental datasets.

REFERENCES

[1] G. Bello-Orgaz, J. J. Jung, and D. Camacho, "Social big data: Recent achievements and new challenges," *Information Fusion*, vol. 28, pp. 45 – 59, 2016.

[2] H. A. L. Thi, X. T. Vo, and T. P. Dinh, "Feature selection for linear svms under uncertain data: Robust optimization based on difference of convex functions algorithms," *Neural Netw.*, vol. 59, pp. 36–50, 2014.

[3] K.-J. Wang, K.-H. Chen, and M.-A. Angelia, "An improved artificial immune recognition system with the opposite sign test for feature selection," *Knowledge-Based Systems*, vol. 71, pp. 126–145, 2014.

[4] R. H. W. Pinheiro, G. D. C. Cavalcanti, and T. I. Ren, "Data-driven global-ranking local feature selection methods for text categorization," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1941–1949, 2015.

[5] Y. Liu, K. Wei, K. Kirchhoff, Y. Song, and J. Bilmes, "Submodular feature selection for high-dimensional acoustic score spaces," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Conference Proceedings, pp. 7184–7188.

[6] S. Maldonado, R. Montoya, and R. Weber, "Advanced conjoint analysis using feature selection via support vector machines," *European Journal of Operational Research*, vol. 241, no. 2, pp. 564–574, 2015.

[7] W.-Y. Deng, D. Liu, and Y.-Y. Dong, "Feature selection and classification for high-dimensional incomplete multimodal data," *Mathematical Problems in Engineering*, vol. 2018, p. 9, 2018.

[8] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[9] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine learning: proceedings of the eleventh international conference*, 1994, Conference Proceedings, pp. 121–129.

[10] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[11] M. Soheili, A.-M. Eftekhari-Moghadam, and M. Dehghan, "Statistical analysis of the performance of rank fusion methods applied to a homogeneous ensemble feature ranking," *Scientific Programming*, vol. 2020, 2020.

[12] R.-J. Palma-Mendoza, L. de Marcos, D. Rodriguez, and A. Alonso-Betanzos, "Distributed correlation-based feature selection in spark," *Information Sciences*, vol. 496, pp. 287–299, 2019.

[13] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Analytics*, vol. 1, no. 1, p. 9, 2016.

[14] M. Soheili and A. M. E. Moghadam, "Feature selection in multi-label classification through mlqpfs," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, 2016, Conference Proceedings, pp. 430–434.

[15] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey, "Effective global approaches for mutual information based feature selection," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 512–521.

[16] H. Venkateswara, P. Lade, B. Lin, J. Ye, and S. Panchanathan, "Efficient approximate solutions to mutual information based global feature selection," in *2015 IEEE International Conference on Data Mining*, 2015, pp. 1009–1014.

[17] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[18] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz, "Quadratic programming feature selection," *Journal of machine learning research : JMLR.*, vol. 11, no. 1, pp. 1491–1516, 2011.

[19] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, F. Herrera, and A. Alonso-Betanzos, "Fast-mrmr: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data," *International Journal of Intelligent Systems*, vol. 32, no. 2, pp. 134–152, 2017.

[20] R.-J. Palma-Mendoza, D. Rodriguez, and L. de Marcos, "Distributed relieff-based feature selection in spark," *Knowledge and Information Systems*, vol. 57, no. 1, pp. 1–20, 2018.

[21] D. Harnie, M. Saey, A. E. Vapirev, J. K. Wegner, A. Gedich, M. Steijaert, H. Ceulemans, R. Wuyts, and W. De Meuter, "Scaling machine learning for target prediction in drug discovery using apache spark," *Future Generation Computer Systems*, vol. 67, pp. 409–417, 2017.

[22] I. Mavridis and H. Karatza, "Performance evaluation of cloud-based log file analysis with apache hadoop and apache spark," *Journal of Systems and Software*, vol. 125, pp. 133–151, 2017.

[23] P. Daniel, S. R.-G. Sara del Río, J. M. B. Isaac Triguero, and F. Herrera, "Evolutionary feature selection for big data classification: a mapreduce approach," *MATHEMATICAL PROBLEMS IN ENGINEERING*, p. 11, 2015.

[24] C. Eiras-Franco, V. Bolón-Canedo, S. Ramos, J. González-Domínguez, A. Alonso-Betanzos, and J. Touriño, "Multithreaded and spark parallelization of feature selection filters," *Journal of Computational Science*, vol. 17, Part 3, pp. 609–619, 2016.

[25] Z. C. Dagdia, C. Zarges, G. Beck, H. Azzag, and M. Lebbah, "A distributed rough set theory algorithm based on locality sensitive hashing for an efficient big data pre-processing," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, Conference Proceedings, pp. 2597–2606.

[26] Z. C. Dagdia, C. Zarges, G. Beck, and M. Lebbah, "A distributed rough set theory based algorithm for an efficient big data pre-processing under the spark framework," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, Conference Proceedings, pp. 911–916.

[27] S. Ramírez-Gallego, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, A. Alonso-Betanzos, and F. Herrera, "An information theory-based feature selection framework for big data under apache spark," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1441–1453, 2018.

[28] M. Soheili and A. M. Eftekhari-Moghadam, "Dqpfs: Distributed quadratic programming based feature selection for big data," *Journal of Parallel and Distributed Computing*, vol. 138, pp. 1–14, 2020.

[29] Y. Prasad, K. K. Biswas, and P. Singla, "Scaling-up quadratic programming feature selection," in *Proceedings of the 17th AAAI Conference on Late-Breaking Developments in the Field of Artificial Intelligence*. AAAI Press, 2013, Conference Proceedings, pp. 95–97.

[30] P. Hanchuan, L. Fuhui, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.