Distributed Ensemble Feature Selection Framework for High–Dimensional and High-Skewed Imbalanced Big Dataset

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

2021 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-7281-9048-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSCI50451.2021.9659937

***Abstract*—The class-imbalance problem emerges when the class labels of a dataset have a skewed distribution. In this circumstance, the instances belonging to one class, which is exactly the principal purpose, are dominated thoroughly by the instances belonging to other classes. In recent years, feature selection for high-dimensional imbalanced data has become attraction research scope. This technique concerns selecting an informative feature set to improve the accuracy of the classification model. Moreover, as a subcategory of feature selection, the feature ranking technique has been deliberated to cope with high-dimensional datasets in the last decade. On the one hand, most traditional feature selection methods are not scalable, which is critical to cope with large-scale datasets. On the other hand, scalability is an intrinsic characteristic of the ensemble learning approach. This paper proposes a Distributed Ensemble Imbalanced feature selection framework, called DEIM, to deal with big imbalanced datasets.**

The DEIM, at first, transforms default data partitions to representative partitions in a single pass. Second, it applies a feature ranking method in a bagging approach upon each partition independently. Finally, It fuses intermediate feature rankings in a stacking strategy. In this paper, two traditional feature ranking algorithms, ReliefF and QPFS, are plugged into DEIM. Therefore, two methods DEIM-Relief and DEIM-QPFS, are produced. Experiments are accomplished on three big imbalanced datasets and upon a computer cluster. The empirical study depicts that the produced methods are scalable. Also, they have lower execution times, and their final results can induce better classification models than DiReliefF and DQPFS.

***Index Terms*—Scalable Feature Selection, Distributed Ensemble Learning, Imbalanced Big Data Set**

I. Introduction

With advances in digital technology and the vast diffusion of IoT devices in the whole world, the data generation rate has been increased tremendously, such that it has not been imaginable earlier. Complexity and huge volume are two prominent properties of this phenomenon, which is known as Big Data. Such massive data include great valuable insight so that applying data discovery algorithms to acquire that has become crucial. Most traditional data discovery algorithms are not scalable. They have been proposed to execute in a centralized computing environment where data can be stored, loaded, and processed by only a single machine. This restriction is not satisfactory in the Big Data era. Therefore, to analyze large-scale datasets, an appropriate data discovery algorithm should be scalable and distributable.

Many applications in the Big Data era, including anomaly detection, and disease diagnosis, produce imbalanced datasets for corresponding main classification tasks [1]. In an imbalanced dataset with a binary class, the number of instances which belong to one class, the majority class, absolutely dominates the number of instances that belong to another class, the minority class. Typically, most classifier algorithms that confront with imbalanced dataset neglect the instances of the minority class [2]. Consequently, the classifier's model has a tremendous general accuracy while it has a weak performance to classify the instances of the minority class. Meanwhile, in most studies, the minority class has more sensitivity and priority to learn correctly than the majority class. Moreover, the complexity of the class imbalance problem is increased dramatically when the dataset is high dimensional.

Several approaches have been proposed to address the imbalanced dataset classification problem such that they can be categorized into four main groups, Data-Level, Algorithm-Level, Cost-sensitive, and Classifier ensembles [3]. Most methods in these approaches may not work well when substantial data dimensions. In this circumstance, the feature selection methods can be constructive for facing imbalanced datasets [4].

Selecting a subset of original features such that the redundancy among them becomes minimum, and the relevancy between them and the class label becomes maximum is the primary objective of the feature selection technique [5]. Two main subdivisions have been introduced regarding feature selection algorithms' outcomes, including feature subset selection (FSS) and feature ranking (FR). FSS methods' outcome is a subset of features that can collectively keep predictive capability the same as original features. In comparison, FR methods' outcome is a ranking of features in a manner that the more informative features will be stood in better (earlier) positions [6]. Also, per learning model dependency, these algorithms can be categorized into three groups: Wrapper, Embedded, and

Filter. In opposite to other groups, the algorithms belonging to the filter category have neither dependency on a classification model. Therefore they have better performance and can afford outcomes with a better generalization, which are worthwhile in the Big Data environment [7].

In recent years, the ensemble learning approach has been mainly appealed in classification algorithms, while potentially it can be an efficient approach in other data mining methods, including feature selection as well [8]. The main idea of ensemble learning is that combining several models instead of a single model can obtain more accurate results [8]. In addition to hoping for a better result, scalability is an innate characteristic of ensemble learning because independent computing nodes can generate base models. This characteristic causes this approach would be suitable in the Big Data environment.

Apache Spark is the most famous distributed computing platform that can process massive datasets in memory. Apache Spark utilizes the Map-Reduce programming paradigm for parallel massive data processing over a computer cluster [9]. Also, it applies an efficient data abstraction named resilient distributed dataset (RDD), a fault-tolerant collection of elements that can be operated on in parallel. The proposed framework is implemented based on the Apache Spark computing model.

In this paper, a Distributed Ensemble Imbalanced feature selection framework, called DEIM henceforth, is presented to cope with big imbalanced datasets. In high skewed big imbalanced datasets, It is most likely that some data partitions do not have any instances belonging to the minority class. In this situation, the data partitions are not representative. Consequently, the informative features of data partitions to recognizing instances of the minority class would be lost. To this matter, DEIM utilizes a novel and approximated method to make representative data partitions only in a single pass. Next, DEIM applies a feature ranking method on a bag of random under-sampling datasets in each data partition. Finally, the intermediate feature rankings are fused in a stacking approach in two separated levels. In the empirical study, two powerful and famous feature ranking algorithms, ReliefF [10] and QPFS [11], are plugged as the base rankers into the DEIM. Consequently, two algorithms, called DEIM-QPFS and DEIM-Relief, are produced to cope with three large and high-skewed imbalanced datasets. The produced methods belong to the filter category because the base ranking methods plugged in DEIM belonging the filter category. The experimental results are compared with DiReliefF [12] and DQPFS [9]. The main contributions with this work are as follows:

- Contributing feature ranking methods in the MLIB library as a famous library based on Apache Spark computing model. To this matter, a distributed ensemble imbalanced feature ranking framework is proposed, which is presently less investigated.
- Proposing a new pipeline as a general feature ranking framework for coping with imbalanced large-scale datasets.
- Experimental study of accuracy and scalability factors of the proposed framework.

The rest of the paper is arranged as follows. Section II considers the related works of feature selection methods. In section III, the main idea of the DEIM framework and two produced methods, DEIM-QPFS and DEIM-Relief, are explained extensively. Section IV describes the implementation of the DEIM framework. The experimental study will be explained in section V. Eventually, section VI raises concluded notes and prominent issues for future works.

II. Related Works

This section investigates some feature selection methods that are adaptable to the distributed environment or proper to coping with imbalanced datasets.

Maldonado et al. considered feature selection and classification task issues simultaneously for binary, small sample size, and imbalanced datasets [4]. They proposed a set of methods for ranking features resting on a backward elimination approach. The DBFS method proposed by Beigi et al. to tackle the small sample size in imbalanced datasets can also be notified [13]. The DBFS method applies the distribution of features over classes to explore the merit of a feature. Both mentioned papers under-considered imbalanced high dimensionality of the given datasets, but the scalability has been neglected.

Hongmei et al. proposed a feature selection algorithm, called RSFSAID, for imbalanced data employing neighborhood rough set theory [14]. The proposed method defines a discernibility matrix and applies particle swarm optimization to determine the optimized parameters in the algorithm. The experimental study is performed on public data sets and depicts that the RSFSAID algorithm can improve the classification performance of imbalanced data compared to four other algorithms; however, this study ignored the scalability.

Viegas et al. [15] proposed an evolutional feature selection method based on genetic programming to deal with high dimensional skewed datasets. In their approach, four feature selection metrics are utilized and their goal was to demonstrate the feasibility of their combination. As another feature selection method based on the evolutional approach, the paper of Susana et al. [16] can be noticeable, but both of these papers have used a classification model as an evaluation function, which would be highly time-consuming in coping with large-scale datasets.

In addition to the above-mentioned articles, some other papers proposed scalable feature selection methods, whereas their methods did not concern the imbalanced datasets. Ramirez et al. published three papers about scalable feature selection. The first work introduced a distributed version of mRMR, the traditional feature selection method, called fast-mRMR [17]. The authors considered the speed-up measure for performing experimental studies, and they inform that their proposed method has proper scalability. The second work introduced a feature selection framework that provides a scalable approach for a family of information-theory-based methods [18]. The researcher concluded that the proposed

framework could handle large-scale and ultra-high dimensional datasets in this work. The third work proposed a redesign version of classical feature selection, ReliefF, called BELIEF [19]. The proposed method utilizes a scalable approach for computing feature scores then eliminates the redundant features by applying a novel measure called mCR. The experiments depicted that their proposed method has smooth scalability. These works were implemented relied on the Apache Spark computing framework, and the experiments were carried out on large-scale datasets, whereas the imbalanced dataset was not considered.

Furthermore, two other scalable feature selection methods were proposed by Mendoza et al. as well. As the first work, they introduced a new version of the traditional ReliefF method, named DiRelief [12]. The experimental results show that the proposed method is scalable, can handle large-scale datasets, and has a lower execution time than the traditional version. The second work, relying on the classical correlation-based feature selection (CFS), proposed two distributed algorithms applying the horizontally and vertically data partitioning, named DiCFS-hp and DiCFS-vp [20]. Their methods were implemented based on the Apache Spark programming model. The experimental studies were performed over a cluster of computers to confront large-scale datasets, although coping with imbalanced datasets was not considered in these studies.

Soheili et al. [9] proposed a feature selection method named DQPFS, a scalable and redesigned version of the traditional method, quadratic programming feature selection, QPFS. Experiments illustrated that the proposed method affords good scalability for coping with large-scale datasets. This capability leads to lower execution time than the traditional version, whereas the outcome was not destroyed. Moreover, the author's other work investigated the various rank combination methods in an ensemble feature selection approach [21]. Their proposed method focused on scalability and combining feature rankings, whereas coping with imbalanced datasets was not investigated.

In summary, some feature selection methods have been introduced in prior works to cope with imbalanced datasets or adapt to distributed environments. However, no research proposes a method to execute in a distributed environment and handle imbalanced datasets, to the best of our knowledge.

III. DEIM Explanation

This paper proposes a distributed and scalable ensemble feature selection framework for coping with big imbalanced datasets. As Fig. 1 illustrates, the main idea of the DEIM framework has five steps. Firstly, performing uniform class label distribution to produce representative data partitions. Secondly, making balanced sub-datasets of representative data partitions. Thirdly, processing each balanced sub-dataset locally and independently using a feature ranking algorithm and then producing corresponding intermediate feature ranking. Eventually, fusing the intermediate feature rankings to generate the final result by applying fusion methods in two steps, four and five. Processing a large dataset as some sub-datasets independently by separate computing nodes reduces communication cost and the execution time of the feature selection algorithm that is a proper advantage in a distributed environment. Moreover, fusing the intermediate feature rankings to produce the final result can cause a more accurate and more stable final result that those are an expected benefit of the ensemble learning. More details of this idea will be explained in the below sub-sections.

A. Making Uniform Label Distribution

Generally, the DEIM framework has two options to make sub-datasets for providing data diversity: first, sampling of the distributed dataset, second, assuming each default data partition as a sub-dataset. The first option has some disadvantages, including that it needs to repeatedly explore whole data partitions stored in the distributed file system (DFS) to make required sub-datasets. Moreover, the generated sub-datasets have to be processed as a sequential. The second option has a disadvantage; there is no assurance that each default data partition in the DFS is a representative partition. In this paper, the representative partition refers to a data partition in which the class label's distribution is approximately the same as the class label's distribution of the whole given big dataset. This approach to making representative partitions is similar to what happens in the stratified sampling method [4]. Addressing the second option's weakness, all instances that belong to a specific class label should be dispatched approximately equal in all data partitions to make representative data-partitions.

As an example, the data partitions of a big imbalanced dataset are represented in Fig. 2. The number of data partitions is 32, such that before balancing the distribution of class labels, data-partitions 1 to 29 have only instances that they belong to the majority class label. Data partition number 30 has a few instances of the minority class label as well, but data-partition number 31 only has instances that belong to the minority class label. After balancing the distribution of class labels, all of the data partitions have instances of both class labels such that the rate of the majority class labels to minority class labels is approximately equal in all partitions. Consequently, the main objective of this step is to make a uniform class label distribution as the same as Fig. 2b. This idea is depicted in the first phase of the proposed framework schema in Fig. 1.

B. Making Balanced Dataset by Under Sampling

After completing the first phase, each data partition of the given big imbalanced dataset is representative. Therefore, each data partition is imbalanced like the whole given dataset, and each partition is processed independently by a worker node In phase two, each data partition is undersampled b times, such that the b is set as an input parameter. Notably, in the undersampling method, the majority class is sampled to the number of instances of the minority class. Then the sampled instances belonging to the majority class are blended with all instances belonging to the minority class to making a balanced sub-dataset. In phase three, the produced balanced dataset is

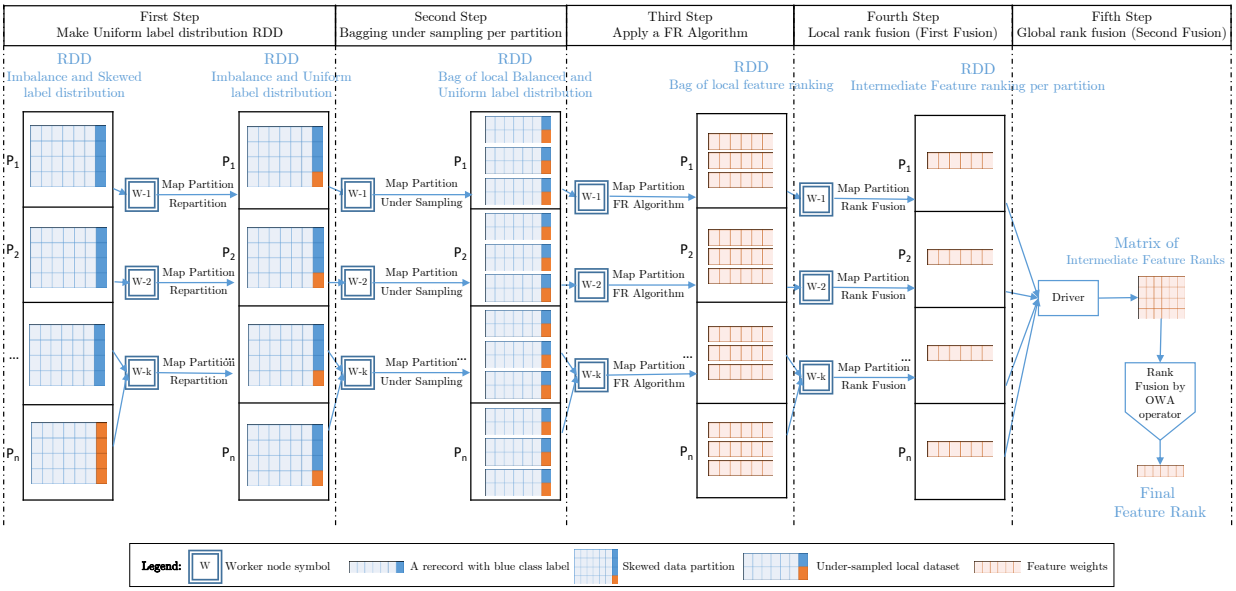


Fig. 1: The schema of the DEIM framework

(a) Before balancing. (b) After balancing.

Fig. 2: Data partition schematic of an imbalanced big dataset.

processed by a feature ranking algorithm, and then a feature ranking result is generated. This matter is repeated b times in a sequential approach for each data partition.

C. Base Feature Ranking Methods

The DEIM framework can utilize an arbitrary feature ranking algorithm, such as Fisher, and Gini Index [21], as the base learner in phase three. Nevertheless, algorithms that belong to the filter category would be better than others because they have lower computational costs and more generality [22]. In this paper, two popular and traditional feature ranking methods, including QPFS [11] and ReliefF [10], are plugged into the DEIM that causes produce two distributed ensemble feature ranking methods called DEIM-QPFS and DEIM-Relief. Consequently, the experimental study to investigate the performance of the DEIM framework is performed based on these produced methods.

D. Fusing the Intermediate Feature Rankings

At the end of phase three, each data partition is transformed to the b number of intermediate feature rankings, which should be reduced to a final feature ranking by applying a rank fusion method. The rank fusion is known by various names, such as Rank Combination and Rank Aggregation as well, and some methods such as Borda, Kwik-Sort, and Stuart are introduced in this field [21]. Due to some advantages, the proposed framework applies an Ordered Weighted Averaging (OWA) operator to reduce the intermediate feature rankings.

These advantages are as follows. First, The OWA operators are easy-used by utilizing a weight vector. Second, there is no tie in the result generated by an OWA operator though that is a usual issue in some classical rank fusion methods such as arithmetic Min and Max. As the third and the most important benefit, the OWA affords a vast and parameterized mechanism for finding a proper model to interact among rankings [23].

OWA operator was introduced by Yager firstly [24], and it is a well-established method for information aggregation. Formally, the OWA operator $O_w = \Re^n \rightarrow \Re$ associates a weight vector $W = (w_1, w_2, w_3, \ldots, w_n)^T$ such that $w_i > 0, \Sigma_{i=1}^{n} w_i = 1$. To the set O_w of n criteria as $O_w(a_1, a_2, a_3, \ldots, a_n) = \Sigma_{i=1}^{n} w_i b_i$ such that the b_i is the ith largest of $(a_1, a_2, a_3, \ldots, a_n)$. It is evident that the output of the O_w strongly depends upon the weight vector W. Therefore if the vector W is set to $(1, 0, \ldots, 0)$, the result of the O_w is equal to the largest number of $(a_1, a_2, a_3, \ldots, a_n)$, namely $O_w = \max$, also when the weight vector W is set to $(0, \ldots, 0, 1)$, it would be $O_w = \min$.

Using a probability density function is one of the strong approaches for determining OWA operator weights [25]. $Dispersion(entropy)$ and $Orness$ are two major properties of a OWA weight vector that they are applied to generate weight vectors [24]. The former property, which is defined $Dispersion(W) = -\sum_{i=1}^{n} w_i \ln w_i$, measures which all aggregates are equally used, whereas the later property, which is defined $Orness(W) = \frac{1}{n} \sum_{i=1}^{n} (n-i)\, w_i$, measures to which the OWA is like an OR operation [26].

In this paper, an OWA operator is applied in such a way its weights vector is generated by the OWG method proposed by Lenormand in 2018 [24]. This method is based on the

probability density functions of truncated normal distributions. By using the OWG method, OWA weights can be generated automatically according to certain values of three input parameters, n, $risk$, and $trade\text{-}off$. The n is equal to the number of criteria or length of the vector of weights, the $risk$ is equal to the $Orness$ property, and the $trade\text{-}off$ is the distance between the weight vector O_w of an OWA operator and the uniform weight vector $\left(w_i = \frac{1}{n}\right)$. Note that the uniform weight vector has the maximum value of the $Dispersion$ property, 1. The OWG method has assumed a default value for the $trade\text{-}off$ parameter as $4 \times risk \times (1 - risk)$ for generating an accurate result.

IV. DEIM Implementation

In this section, the phases of the DEIM framework explained in the previous section will be detailed in some pseudocodes. It is worth mentioning that the DEIM is implemented relied on the Apache Spark computing framework and the Map-Reduce paradigm. The main procedure of the DEIM is detailed by Algorithm 1. The DEIM's phases in Algorithm 1 are, (i) Making Uniform Label Distribution, in line 3, (ii) Bagging Under Sampling, in lines 6–8. (iii) Applying a Feature Ranking Algorithm, line 9. (iv) Applying Local Rank Fusion, in line 11. (v) Applying Global Rank Fusion, in line 13.

In Algorithm 1, the $DS \in \mathbb{R}^{n \times d}$ refers to the given dataset with n instances and d dimensions. Furthermore, p refers to the number of new data partitions with uniformed class label distribution, and it will be passed to Algorithm 2. In another perspective, the parameter p can be inferred as the number of data subsampling applied in the ensemble learning approach. Input parameter b refers to the number of local balanced datasets to be generated in each data partition. Input parameter RA refers to the feature ranking algorithm applied in each local balanced dataset independently. Input parameters r_1 and r_2 refer to two risk parameters applied to the rank fusion method, and they will be passed to Algorithm 3.

As the first phase of Algorithm 1, the given dataset is transformed to a uniform class label distribution dataset by applying the $UCDL$ method of Algorithm 2. To this aim, each data partition tries to dispatch its instances that belong to the same class label, among all new data partitions almost equally. Therefore, the default data partition determines a new data partition index for each instance. For this purpose, an instance with class label $label$ is dispatched to a new data partition with the smallest number of instances with class label $label$. If there are multiple new data partition candidates for a specific instance, one is selected randomly. This dispatching policy is repeated for all instances in all default data partitions independently.

To implement the dispatching policy in Algorithm 2, a matrix, $labelParts \in \mathbb{N}^{nc \times p}$ is applied to keep the frequency of the same class label instances dispatched among new data partitions. In this matrix, the number of rows is equal to the number of unique class labels, nc, of the given dataset, and the number of columns is equal to the number of new data partitions,p. Each default data partition has its own $labelParts$ matrix. For each instance of a default data partition, a new data partition index is determined based on the $labelParts$ matrix and explained dispatching policy.

Therefore, each instance in each default data partition transforms into a $\langle key, value \rangle$ tuple such that the key is set to a new partition index, and the $value$ is set to the instance. Finally, the hash partitioning method is applied to perform new data partitions. The hash partitioning method shuffles whole tuples among all data nodes in such a way the entire tuples in all data partitions that have the same key will be placed in the same data partition. The result of partitioning by applying the hash partitioning method is a dataset with uniformed class

Algorithm 1: DEIM - Main Procedure

Input : $DS \in \mathbb{R}^{n \times d}$ // the given dataset
Input : $p \in \mathbb{N}$ // the number of new data Partitions
Input : $b \in \mathbb{N}$ // the number of Bagging sampling
Input : RA // the feature Ranking Algorithm
Input : $r_1 \in \mathbb{R}$ // the first Risk parameter
Input : $r_2 \in \mathbb{R}$ // the second Risk parameter
Output: $RR \in \mathbb{N}^d$ // the feature Ranking Result

1 $FRM_{local} = \text{Matrix}\,[b, d]$ // the feature rankings matrix
2 $FRM_{global} = \text{Matrix}\,[p, d]$ //
3 $DS_{ud} = \text{UCLD}\,(DS, p)$ // Algorithm 2
4 **forall** $partition \in DS_{ud}$ **do in parallel**
5 **for** $i = 0$ **to** b **do**
6 $(ds_{minor}, ds_{major}) \leftarrow$ Split $partition$ into two sub datasets, minor class label and major class label.
7 $ds_{sub-major} =$ Subsampling of ds_{major}
8 $ds_{sub-balanced} = ds_{minor} \bigcup ds_{sub-major}$
9 $FRM_{local}\,[i, :] = \text{FRA}\,(ds_{sub-balanced})$
10 **end**
11 $FRM_{global}\,[partition.index, :] = \text{OWAF}\,(FRM_{local}, r_1)$ // Algorithm 3
12 **end**
13 $FR = \text{OWAF}\,(FRM_{global}, r_2)$ // Algorithm 3
14 **return** FR

Algorithm 2: UCLD- Uniform Class Label Distribution

Input : $DS \in \mathbb{R}^{n \times d}$ // the given dataset
Input : $p \in \mathbb{N}$ // the number of new data Partitions
Output: $DS_{ud} \in \mathbb{R}^{n \times d}$ // the Uniformed class label Distribution dataset

1 $nc \leftarrow$ Count distinct value of $DS\,[:, d-1]$ // the Number of Class label
2 $DS_t =$
3 **forall** $partition \in DS$ **do in parallel**
4 $labelParts = \text{Matrix}\,[nc, p]$
5 **forall** $instance \in partition$ **do**
6 $label = instance\,[d-1]$ // the last column is the class label
7 $parts = labelParts\,[label, :]$
8 $index = \text{findSmallest}\,(parts)$ // find the cell index that has smallest value as a new partition index
9 $labelParts\,[label, index] += 1$
10 $\langle index, instance \rangle$
11 **end**
12 **end**
13 $DS_{ud} = DS_t.\text{partitionBy}\,(\text{HashPartitioner}\,(p))$
14 **return** DS_{ud}

label distribution such that each of its data partitions is a proper representative. Algorithm 2 explains the uniform class label distribution method in detail.

As mentioned before, each representative data partition provided in the first phase of Algorithm 1 is processed independently by a worker node. Next, in phase two of Algorithm 1, a local and balanced sub-datasets is generated in a way explained before in section III-B. Then, in phase three of Algorithm 1, the local and balanced sub-dataset is processed by a feature ranker algorithm, and then the corresponding feature ranking is produced. Consequently, after repeating phases two and three by b times, each representative data partition has been transformed into a matrix of intermediate feature ranking results, $FRM_{local} \in \mathbb{N}^{b \times d}$.

The intermediate feature rankings are reduced to a final feature ranking by applying the OWAF rank fusion method in two levels. At the first level of fusion, in phase four of Algorithm 1, each data partition's intermediate feature ranking results, whose number is equal to b, are reduced to a single ranking of features independently. Consequently, a global feature ranking matrix, $FRM_{global} \in \mathbb{N}^{p \times d}$, is produced at the end of phase four. At the second level of fusion, in phase five of Algorithm 1, the FRM_{global} matrix, whose number of rows is equal to the number of new data partitions, is reduced to a single final ranking of features as the final result. Therefore, two risk parameters should be set as input parameters for applying two-level rank fusion in a stacking approach.

Algorithm 3 details how a feature rankings matrix is fused for making a final ranking. In this algorithm, the input parameter $FRM \in \mathbb{N}^{n \times d}$ refers to a matrix whose rows are feature rankings. It is worth mentioning that OWG is a weight vector generator method presented in [24], and it receives three input parameters $risk$, $trade\text{-}off$, and n. The $risk$ is set by input parameter r, the $trade\text{-}off$ is set to a default value by t local variable, and the n is the number of criteria that in Algorithm 3 refers to the row number of FRM matrix. Finally, a single feature ranking result, RL, is returned by the algorithm.

Algorithm 3: OWAF - OWA Rank Fusion

Input : $FRM \in \mathbb{N}^{n \times d}$ // the **Feature Ranking Matrix**
Input : $r \in \mathbb{N}$ // the Risk of weight vector
Output: $RL \in \mathbb{N}^{d}$ // the Ranking Result

$t = 4 \times r \times (1 - r)$ // the default value for *trade-off*
$FP = \text{Matrix}\,[d, n]$ // a matrix of Feature Positions
$FW = \text{Array}\,[d]$ // an array of Feature Weights
for $i = 0$ *to* $(n - 1)$ **do**
  for $j = 0$ *to* $(d - 1)$ **do** $FP\,[FRM\,[i, j]\,, i] = j$
end
$W = OWG\,(r, t, n)$ // a weight vector based on [24]
for $i = 0$ *to* $(d - 1)$ **do** $FW\,[i] = FP\,[i, :] \times W^{T}$
$RL = \text{Order}\,(FW)$
return RL

V. Experimental Study

Two traditional feature ranking algorithms, QPFS and ReliefF, are plugged in the DEIM framework as the base learners to perform the experimental study. Consequently, two algorithms, called DEIM-QPFS and DEIM-Relief, are produced. Two produced methods are executed on three experimental datasets, and then their results are collected. Next, the various Gaussian Naive Bayes and Random Forest classification models are induced based on the selected features. Accordingly, the outcomes of the classifier are gathered as final results by applying a 5-fold cross-validation method. Note that two ad-hoc parameters, the number of samples and neighbors, are set to 20 in DiReliefF and DEIM-Relief algorithms. Furthermore, the DQPFS and DEIM-QPFS algorithms utilized information theory. Thus they need to transform continuous features into categorical features. Therefore, continuous features are quantized into ten equal width bins.

TABLE I: The properties of datasets used in experiments.

#	Dataset name	#Instances	#Features	Minor:Major	Format	#Class
1	ECBDL (sampled)	2000000	631	(98:2)	ASCII	2
2	FD (sampled)	2000000	900	(90:10)	Binary	2
3	OCR (sampled)	1800000	1156	(98:2)	Binary	2

For reproducibility objectives, the DEIM framework code has been uploaded to a repository[1].

A. Datasets, and Cluster

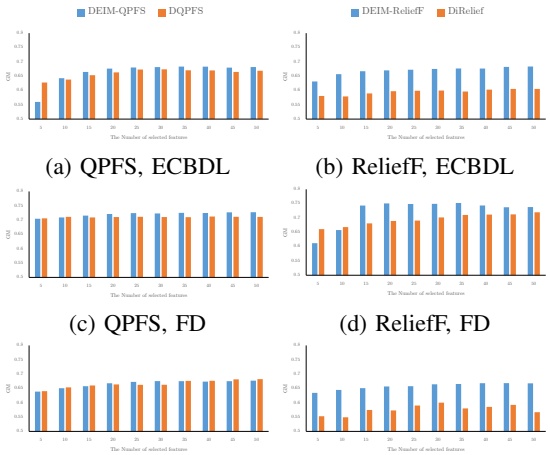
Two produced methods, DEIM-QPFS and DEIM-Relief, are executed on three well-known, high dimensional, and high-skewed imbalanced datasets, ECBDL[2], FD, and OCR[3], whose number of instances and features are large simultaneously. Note that the OCR is originally a balanced dataset, so instances belonging to the negative class are sub-sampled to make it an imbalanced dataset. Table I shows the characteristics of the experimental datasets. The experimental study is performed on binary class datasets, whereas theoretically the proposed framework can be applied in multi-class datasets as well. The experiments are performed upon a cluster with 8-nodes configured based on the master/slave architecture. All nodes have two-core processors, 4 gigabytes RAM, and the Apache Hadoop 2.7 and the Apache Spark 2.3 are installed.

B. Assessment Criteria and Results

In this section, some criteria, such as Classification Outcomes, Execution-Time, and Speed-Up, are considered to assess the performance of the two produced methods, DEIM-QPFS and DEIM-Relief. Moreover, a deeply experimental study compares the experimental results with DQPFS [9] and DiReliefF [12]. DQPFS and DiReliefF are two distributed feature selection algorithms published recently. Thus they are attractive algorithms to study the performance of the produced methods.

1) Geometric Mean (GM): The classification algorithm's accuracy is profoundly misleading for a classification task in the imbalanced domain because minority classes are insignificant in overall accuracy. Consequently, the majority class's performance can dominate the weak performance of the

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



(a) QPFS, ECBDL (b) ReliefF, ECBDL

(c) QPFS, FD (d) ReliefF, FD

(e) QPFS, OCR (f) ReliefF, OCR

Fig. 3: The GM results, DEIM-QPFS vs DQPFS, and DEIM-Relief vs DiReliefF, by applying the Naive Bayes classifier.

(a) QPFS, ECBDL (b) ReliefF, ECBDL

(c) QPFS, FD (d) ReliefF, FD

(e) QPFS, OCR (f) ReliefF, OCR

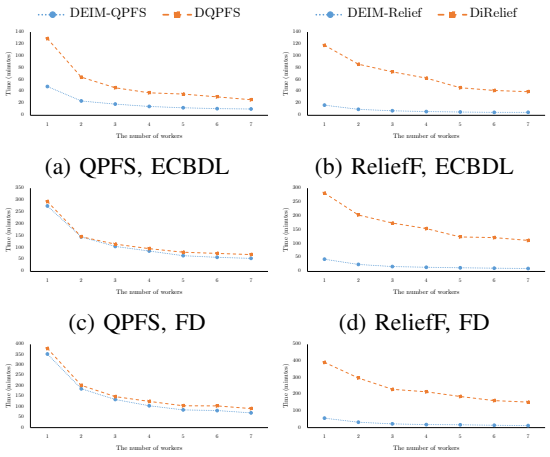
Fig. 4: The GM results, DEIM-QPFS vs DQPFS, and DEIM-Relief vs DiReliefF, by applying the Random Forest classifier.

minority class [1]. Therefore, using a measure that investigates the performance of minority and majority classes altogether is essential. The GM measurement aims to maximize the accuracy of each one of the two minority and majority classes simultaneously, and it has been used widely in literature [1], [3]. This measure is defined as $GM = \sqrt{sensitivity \times specificity}$ where $sensitivity = \frac{TP}{TP+FN}$, and $specificity = \frac{TN}{FP+TN}$. Note that TP, TN, FP, and FN refer to the true-positive, true-negative, false-positive, and false-negative, respectively, in a confusion matrix for a binary classification task. Moreover, to compare the best configurations of the first and second risk parameters of the produced method are selected.

As Fig. 3 and Fig. 4 illustrate, the produced methods, DEIM-QPFS and DEIM-Relief, have better or comparable GM results than the DQPFS and DiReliefF in most experiences (115/120). In other words, just in two experiences, five selected features in Fig 3a and Fig 3d, the produced methods have significantly lower GM results than DQPFS and DiReliefF. Also, these figures depict that the difference between GM results in DEIM-QPFS with DQPFS is bigger than DEIM-Relief with DiReliefF, and this matter is related to the randomized strategy applied in the ReliefF algorithm.

2) Execution Time: The feature ranking algorithm plugged into DEIM strongly affects its execution time. As Fig. 5 illustrates, the execution time of the DEIM-QPFS is significantly more than DEIM-Relief in all experimental datasets, whereas their difference decreases by increasing the number of worker nodes. Another notable point is that the difference in the produced method's execution time is increased by growing the number of experimental dataset's features. Therefore, a large number of dataset features can extend the execution time difference between DEIM-QPFS and DEIM-Relief.

Moreover, the DEIM-QPFS and DEIM-Relief execution time is compared with two distributed algorithms DQPFS and DiReliefF. According to Fig. 5, two produced methods have lower execution times than DQPFS and DiReliefF. Whereas the difference between DEIM-Relief and DiReliefF



(a) QPFS, ECBDL (b) ReliefF, ECBDL

(c) QPFS, FD (d) ReliefF, FD

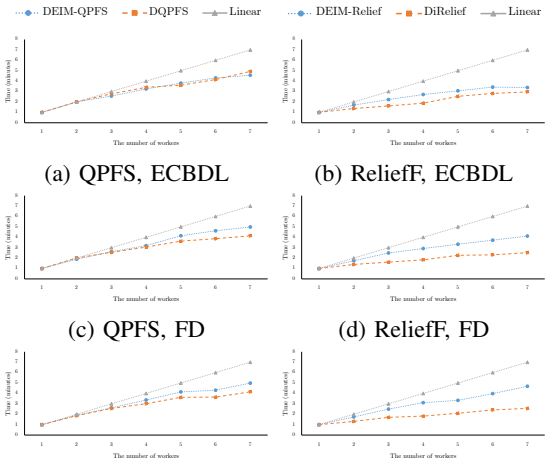
(e) QPFS, OCR (f) ReliefF, OCR

Fig. 5: The execution time, DEIM-QPFS vs DQPFS, and DEIM-Relief vs DiRelief

is more significant than the difference between DEIM-QPFS and DQPFS.

3) Speed-Up: The speed-up as an important scalability measure indicates an algorithm's capability to exploit increasing of worker nodes in order to execution time reduction. The speed-up is computed as $Speed\text{-}Up = \frac{T(n,1)}{T(n,p)}$. Notice that in the equation, $T(n,p)$ refers to the execution time of the algorithm processes the dataset with the n number of instances by utilizing the p number of worker nodes.

As Fig. 6 depicts, both produced methods have a proper speed-up, whereas the DEIM-QPFS has a better performance than DEIM-Relief. In all charts, the speed-up will be decreased by increasing the number of worker nodes. It is typical behavior in distributed algorithms and relates to communication costs. The communication cost will be increased by increasing the number of participating worker nodes. According to Fig. 6, the *speed-up* of DEIM-QPFS and QPFS is similar, and DEIM-Relief is better than DiReliefF in experimental datasets.



(a) QPFS, ECBDL (b) ReliefF, ECBDL

(c) QPFS, FD (d) ReliefF, FD

(e) QPFS, OCR (f) ReliefF, OCR

Fig. 6: The speed-up, DEIM-QPFS vs DQPFS, and DEIM-Relief vs DiReliefF

VI. Conclusion

This paper proposes a distributed and scalable feature ranking framework to cope with high-dimensional and high-skewed imbalanced big datasets, called DEIM.

The DEIM makes representative data partitions from the default data partitions at the first step. Next, the DEIM creates a bag of balanced small datasets on each representative data partition and applies a feature ranking algorithm on each of the small balanced datasets. Finally, it fuses the intermediate results by applying an OWA operator in two levels.

In the DEIM framework, each arbitrary feature ranking algorithm can be plugged as the base learner. In this paper, two popular feature ranking algorithms, QPFS and ReliefF, are utilized. Therefore, two distributed ensemble algorithms, DEIM-QPFS and DEIM-Relief, are produced. The experimental study is performed on a cluster of computers upon three big imbalanced datasets. The experiments' outcomes illustrate that the produced methods are scalable. Moreover, in comparison to distributed versions of the classical algorithms, DQPFS and DiReliefF, produced methods are faster and can produce more suitable outcomes based on GM in most experiments for diverse numbers of selected features. The comparison between the DEIM-QPFS and DEIM-Relief illustrates that the former has a longer execution time and better Speed-Up than the latter.

References

[1] J. Kim, J. Kang, and M. Sohn, "Ensemble learning-based filter-centric hybrid feature selection framework for high-dimensional imbalanced data," *Knowledge-Based Systems*, vol. 220, p. 106901, 2021.

[2] S. del Río, V. López, J. M. Benítez, and F. Herrera, "On the use of mapreduce for imbalanced big data using random forest," *Information Sciences*, vol. 285, pp. 112–137, 2014.

[3] M. Juez-Gil, Álvar Arnaiz-González, J. J. Rodríguez, and C. García-Osorio, "Experimental evaluation of ensemble classifiers for imbalance in big data," *Applied Soft Computing*, vol. 108, p. 107447, 2021.

[4] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Information Sciences*, vol. 286, pp. 228–246, 2014.

[5] Y. Li, T. Li, and H. Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 551–577, 2017.

[6] 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.

[7] V. J. Hodge, S. O'Keefe, and J. Austin, "Hadoop neural network for parallel and distributed feature selection," *Neural Networks*, vol. 78, pp. 24–35, 2016, special Issue on "Neural Network Learning in Big Data".

[8] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information Fusion*, vol. 52, pp. 1–12, 2019.

[9] 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.

[10] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *Machine Learning: ECML-94*, ser. Machine Learning: ECML-94, F. Bergadano and L. De Raedt, Eds. Springer Berlin Heidelberg, 1994, Conference Proceedings, pp. 171–182.

[11] 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.

[12] 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.

[13] M. Alibeigi, S. Hashemi, and A. Hamzeh, "Dbfs: An effective density based feature selection scheme for small sample size and high dimensional imbalanced data sets," *Data and Knowledge Engineering*, vol. 81-82, pp. 67–103, 2012.

[14] H. Chen, T. Li, X. Fan, and C. Luo, "Feature selection for imbalanced data based on neighborhood rough sets," *Information Sciences*, vol. 483, pp. 1–20, 2019.

[15] F. Viegas, L. Rocha, M. Gonçalves, F. Mourão, G. Sá, T. Salles, G. Andrade, and I. Sandin, "A genetic programming approach for feature selection in highly dimensional skewed data," *Neurocomputing*, vol. 273, pp. 554–569, 2018.

[16] S. M. Vieira, J. M. Sousa, and T. A. Runkler, "Two cooperative ant colonies for feature selection using fuzzy models," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2714–2723, 2010.

[17] 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.

[18] 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.

[19] D. López, S. Ramírez-Gallego, S. García, N. Xiong, and F. Herrera, "Belief: A distance-based redundancy-proof feature selection method for big data," *Information Sciences*, vol. 558, pp. 124–139, 2021.

[20] 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.

[21] 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.

[22] 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.

[23] R. R. Yager, "Owa aggregation of multi-criteria with mixed uncertain satisfactions," *Information Sciences*, vol. 417, pp. 88–95, 2017.

[24] M. Lenormand, "Generating owa weights using truncated distributions," *International Journal of Intelligent Systems*, vol. 33, no. 4, pp. 791–801, 2018.

[25] X. Liu, "Models to determine parameterized ordered weighted averaging operators using optimization criteria," *Information Sciences*, vol. 190, pp. 27–55, 2012.

[26] Y. Ouyang, "Improved minimax disparity model for obtaining owa operator weights: Issue of multiple solutions," *Information Sciences*, vol. 320, pp. 101–106, 2015.