applied sciences

MDPI

Article

Data-Centric Solutions for Addressing Big Data Veracity with Class Imbalance, High Dimensionality, and Class Overlapping

Armando Bolívar [1], Vicente García [2], Roberto Alejo [3], Rogelio Florencia-Juárez [2] and J. Salvador Sánchez [4,*]

[1] Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez, Av. del Charro 450 NTE., Ciudad Juárez 32310, Chihuahua, Mexico; al198665@alumnos.uacj.mx
[2] División Multidisciplinaria en Ciudad Universitaria, Universidad Autónoma de Ciudad Juárez, Av. José de Jesús Delgado 18100, Ciudad Juárez 32579, Chihuahua, Mexico; vicente.jimenez@uacj.mx (V.G.); rogelio.florencia@uacj.mx (R.F.-J.)
[3] Division of Postgraduate Studies and Research, Tecnológico Nacional de México, Instituto Tecnológico de Toluca, Av. Tecnológico s/n, Colonia Agrícola Bellavista, Metepec 52149, Estado de México, Mexico; ralejoe@toluca.tecnm.mx
[4] Institute of New Imaging Technologies, Department of Computer Languages and Systems, Universitat Jaume I, Av. de Vicent Sos Baynat s/n, 12071 Castelló de la Plana, Spain
* Correspondence: sanchez@uji.es

Abstract: An innovative strategy for organizations to obtain value from their large datasets, allowing them to guide future strategic actions and improve their initiatives, is the use of machine learning algorithms. This has led to a growing and rapid application of various machine learning algorithms with a predominant focus on building and improving the performance of these models. However, this data-centric approach ignores the fact that data quality is crucial for building robust and accurate models. Several dataset issues, such as class imbalance, high dimensionality, and class overlapping, affect data quality, introducing bias to machine learning models. Therefore, adopting a data-centric approach is essential to constructing better datasets and producing effective models. Besides data issues, Big Data imposes new challenges, such as the scalability of algorithms. This paper proposes a scalable hybrid approach to jointly addressing class imbalance, high dimensionality, and class overlapping in Big Data domains. The proposal is based on well-known data-level solutions whose main operation is calculating the nearest neighbor using the Euclidean distance as a similarity metric. However, these strategies may lose their effectiveness on datasets with high dimensionality. Hence, the data quality is achieved by combining a data transformation approach using fractional norms and SMOTE to obtain a balanced and reduced dataset. Experiments carried out on nine two-class imbalanced and high-dimensional large datasets showed that our scalable methodology implemented in Spark outperforms the traditional approach.

Keywords: big data; class imbalance; high dimensionality; fractional norms; dissimilarity representation

check for updates

Citation: Bolívar, A.; García, V.; Alejo, R.; Florencia-Juárez, R.; Sánchez, J.S. Data-Centric Solutions for Addressing Big Data Veracity with Class Imbalance, High Dimensionality, and Class Overlapping. *Appl. Sci.* **2024**, *14*, 5845. https://doi.org/10.3390/app14135845

Received: 29 May 2024
Revised: 26 June 2024
Accepted: 2 July 2024
Published: 4 July 2024

1. Introduction

Technological advances have enabled the private and public sectors to generate and collect vast amounts of data from various sources. Social media interactions, transactional data, geospatial information, biometric identifiers, and governmental databases are examples of where these data originate from. According to the 11th edition of 'Data Never Sleeps' 241 M emails were sent every minute in 2023 [1]. Consequently, the volume of raw data may reach zettabytes (ZB) in a variety of formats, such as text, audio, image, and video.

The velocity at which data are generated, processed, and analyzed adds another dimension to this data explosion. The International Data Corporation (IDC) and Seagate estimate that by 2025, approximately 163 ZB of data will have been generated, indicating that the volume may have grown tenfold in the last eight years [2]. Therefore, data can be characterized by their volume, variety, and velocity, which is also called the three Vs [3], which define the phenomenon known as Big Data [4].

Analyzing Big Data represents a significant opportunity for companies, governments, and society to extract meaningful insights. This process can help us to gain a competitive edge, enhance decision-making, and drive innovation [3,5,6]. Therefore, Big Data can also be described by its value in obtaining benefits and insights from the analyzed raw data, adding a fourth V-characteristic. In this sense, machine learning and artificial communities have identified a niche opportunity to develop techniques that can discover hidden patterns and predict future values or trends from large amounts of complex data from sectors such as healthcare, cybersecurity, finance, marketing, manufacturing, and smart grids [7–11].

In 2021, Andrew Ng stated that much of the effort in deploying machine learning models has been focused on optimizing or creating new algorithms [12]. This model-centric approach ignores the importance of data quality in achieving the most effective machine learning systems [13]. The veracity of data has been recognized as a critical property for obtaining reliable and actionable insights from large datasets. This fifth V-characteristic deals with ensuring the trustworthiness of captured data, where low data veracity caused by a number of issues, such as data redundancy, inconsistency, and noise, may cause machine learning models to yield inaccurate predictions, biased results, and reduced model performance. As a response to addressing data veracity rather than solely concentrating on the algorithms themselves, it has been proposed that the development and application of machine learning models should be with a data-centric approach, aiming to improve the models' performance using high-quality data obtained from data preprocessing techniques that address several data issues.

One of the data inconsistencies leading to inaccurate behavior in classification models is the significant difference in sample sizes per class in the training dataset. This type of bias, where the classes are not equally distributed, is well-known as the class imbalance problem, which causes trained algorithms to favor the predominant class in their predictions [14,15]. This issue becomes severe when minority classes are of major interest, which means the cost of incorrectly labeling an example of an underrepresented class is very high [16]. Therefore, the data preprocessing step is paramount for transforming raw Big Data through data cleaning, extraction, and transformation, resulting in reduced and cleaned data, also known as smart data [7,9,17].

Solutions to the class imbalance problem can be categorized into two large groups called algorithm-level and data-level approaches. Algorithm-level methods modify existing algorithms to learn from imbalanced datasets. In contrast, data-level solutions focus on balancing the training dataset by either reducing (under-sampling) or increasing (oversampling) the sample sizes of different classes. Although data imbalance has traditionally been seen as the main cause of the poor performance of the classifiers, several studies have shown that other data irregularities, such as class overlap [18] and high dimensionality [19], can also exacerbate the problem, introducing a unique set of challenges. Consequently, a third research line has emerged, combining the strengths of both data-level and algorithm-level techniques.

The class imbalance problem has attracted much attention for many years, leading to the development of numerous solutions addressing biased learning. Data-level solutions have been the most exploited of the three research lines because the proposals can be applied to various problems across different domains. For example, in 2002, Chawla et al. [20] introduced an oversampling technique called the Synthetic Minority Oversampling Technique (SMOTE) to balance datasets by generating synthetic examples of the minority class through interpolation between selected minority instances and its nearest neighbors. By 2019, Kovács [21] had documented approximately 85 variants of SMOTE. In the case of under-sampling techniques for selecting instances of the majority class to be removed, the community has adopted the strategy to filter those examples that can be considered noisy, redundant, or borderline. The simplest method to identify these data types is to analyze the local distribution of the data by computing the k-nearest neighbors [22].

Although the class imbalance problem has been effectively addressed in standard scenarios, Big Data introduces additional challenges where some conventional assumptions no longer hold, making traditional data preprocessing methods infeasible [7]. Therefore, data-level solutions must consider the specialized infrastructure to process large datasets in parallel or distributed systems such as Apache Spark and new programming paradigms like Scala [23]. However, it is possible to find strategies in the literature that have been adapted to be scalable solutions. For example, Basgall et al. [23] proposed SMOTE-Big Data (SMOTE-BD), an implementation in Apache Spark. This oversampling technique is based on an innovative distributed k-nearest neighbor model named kNN-IS that enhances the runtime efficiency of the nearest neighbor search [24].

In Big Data environments, similar to standard problems, data-level solutions are based on the nearest neighbor rule, typically employing metrics such as the Euclidean distance to assess the similarity between one example and another. However, computing distances between data samples in high-dimensional datasets can make them appear nearly identical, thereby complicating the differentiation between samples [25,26]. A dataset with a large number of features relative to the number of samples can lead to the phenomenon known as the curse of dimensionality. In high-dimensional spaces, the volume of the space increases exponentially with the number of dimensions, leading to data sparsity. Therefore, the Euclidean distance loses its discriminatory power, impacting the effectiveness of learning algorithms because distances between instances become less informative [27]. In this paper, we hypothesize that the use of fractional norms, norm-p, can alleviate the effect of the high dimensionality in a well-known data-level solution. In particular, we employ a dissimilarity approach to mapping the dataset into a lower dimensionality, in which the dimensions are defined by vectors measuring pairwise dissimilarities between two examples, commonly given by the Euclidean distance [28,29].

We note that data-centric solutions for class imbalance in Big Data scenarios have been less explored. Additionally, some approaches address data issues in an isolated manner. Therefore, we propose a scalable hybrid approach that combines renowned data-level solutions based on distance metrics. Initially, the dissimilarity approach maps the original feature space representation into a low-dimensional dissimilarity space using a fractional norm. In this transformed space, represented by dissimilarity vectors, we then apply SMOTE to achieve class balance.

We conducted a comprehensive experimental study on nine imbalanced Big Data datasets with high dimensionality to evaluate the proposal's performance. The datasets were adapted to two-class problems, comprising 24,832 features and 21,025 instances. The decision tree employed was taken from Spark's MLib toolkit. The proposal was compared against the application of SMOTE in the original space and in scenarios without any balancing technique. A nonparametric statistical test was used in our analysis to determine whether our proposal statistically outperformed the other methods. In summary, the contributions of this paper are as follows:

- We address the problem of class imbalance in the presence of high dimensionality and class overlap.
- We explore the suitability of fractional norms as an alternative to the problem of Euclidean distance in high-dimensional datasets.
- We employ a dissimilarity-based representation to mitigate high dimensionality and class overlap issues.
- We implemented a hybrid approach in Spark.

The rest of the paper is organized as follows. Section 2 describes the methods and techniques used in this work. Section 3 presents the experimental setup. Section 4 discusses the results. Finally, Section 5 remarks on the main findings and outlines further research.

2. Methods

This section offers a general introduction to fractional norms, SMOTE-DB, and dissimilarity representation.

2.1. Fractional Norms

The use of Euclidean distance faces significant limitations in high-dimensional spaces. High dimensionality relates to the similarity space's separability, geometry, topology, and density. It refers to a high number of input variables in the dataset. As the dimensionality of data increases, the data size grows proportionally, leading to data dispersion that complicates the data analysis [30]. This phenomenon is known as the curse of dimensionality.

Distance concentration and hubness are two phenomena that occur in high-dimensional datasets and have a direct negative impact on the nearest neighbor calculation. On the one hand, distance concentration is the characteristic present in high-dimensional spaces where all points seem to be almost the same distance from other points in that space [31]. This makes using distance metrics to discriminate between similar examples ineffective. Meanwhile, hubness is described by Tomasev et al. [32] as the tendency for specific instances to appear frequently in the list of nearest neighbors of other instances.

Aggarwal et al. [33] discuss how distances based on the norm-p, with fractional coefficients, i.e., Minkowski distance metric, can help mitigate distance concentration in high dimensionality. Formally, the Minkowski distance for $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d(\mathbf{x},\mathbf{y}) := \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \quad for \quad p > 1 \tag{1}$$

where p for the fractional distances or quasi-distances take values between 0 and 1. Consequently, fractional distances are not formally defined as distance metrics due to the violation of the triangle inequality [34].

Although it has been demonstrated that fractional distances can mitigate distance concentration [33], selecting the appropriate p is a complex process [31,34–36].

2.2. Dissimilarity Representation

In machine learning, an instance is traditionally represented mathematically as a vector $\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_m\}$, where each x_i is an attribute or feature $\in \mathbb{R}^m$ that describes a property of the instance. This feature-based representation is widely adopted; however, it has been observed that instances with different class labels could have a very similar vector generating ambiguous regions [18,37]. This phenomenon, known as class overlap, poses a challenge in distinguishing the best class label between different classes [18].

As a solution to this problem, alternative representations have been proposed that transform or encode each instance into a new vector. In a fashion such as using kernels, Pękalska and Duin [28] propose mapping a feature vector into a dissimilarity vector using the Euclidean metric, computed between a feature vector and a representation set. The rationale for adopting the dissimilarity representation is that dissimilarity measures should be minimal for examples of the same class and significant for examples of different classes, facilitating discrimination between classes [29,38]. Experiments by García et al. [39] on credit scoring datasets show that dissimilarity representation provides significantly higher separability between classes than the original feature representation.

In the dissimilarity representation, the attributes are defined through vectors that quantify the similarity between examples and certain elements or prototypes belonging to a designated representation set (R). This set R must include examples from all classes and can be selected using various strategies. For instance, R can be defined as the complete training set, a subset derived through a heuristic mechanism, randomly selected, or even extracted from the test dataset [38]. R was randomly formed for the experiments described in this article, ensuring an equitable representation of examples from each class in the training set. This straightforward strategy of forming R has proven effective in most problems [38].

The mapping from the original feature space to the dissimilarity space is carried out by calculating a distance metric between R and each example in the training and test sets [38]. Thus, the dimensions in the dissimilarity space are represented by distance vectors that reflect the dissimilarity between R and the examples in the feature space.

In other words, if we define the representation set as

$$\mathbf{R} = \{r_1, r_2, ..., r_k\} \tag{2}$$

and the training set as **E**, where e symbolizes an example within the training dataset, then any vector in the dissimilarity space can be denoted as

$$d_e = [d_p(e, r_1), d_p(e, r_2), ..., d_p(e, r_k)] \tag{3}$$

where $d_p()$ is the dissimilarity metric. Therefore, the dissimilarity space corresponding to **E** is defined through the dissimilarity matrix $\mathbf{D_E}$, which encompasses all the dissimilarity vectors d_e [38].

Algorithm 1 shows the dissimilarity mapping procedure implemented in Spark. Likewise, it includes the modification for calculating the dissimilarity using fractional distances.

Algorithm 1: Dissimilarity Representation in Spark

```
/* Input and output files are in LIBSVM format. */
/* Train and test data files in the feature space. */
Data: trainDataFile, testDataFile
/* minClassLabel and majClassLabel are the class labels of the minority and majority classes,
   respectively. The values are used to filter the classes within the algorithm. */
/* rPerClass is the number of instances selected per class to conform the representation set R
   to transform the feature space into the dissimilarity space. An equal number of instances
   are selected for each class. */
/* fractionalDistance is the fractional p used to calculate the distance between samples. The
   value is in the range of $0 < p < 1$. */
Input: minClassLabel, majClassLabel, rPerClass, fractionalDistance
/* dissimilarityTrainFile and dissimilarityTestFile are filenames used to write the dataset
   after the dissimilarity transformation. */
Output: dissimilarityTrainFile, dissimilarityTestFile
/* Read train and test data. */
trainData ← spark.read.libsvm(trainDataFile).repartition();
testData ← spark.read.libsvm(testDataFile).repartition();
/* Get R samples from each class. */
rMinSamples ← trainData.filter(label == minClassLabel).orderBy(rand()).limit(rPerClass);
rMajSamples ← trainData.filter(label == majClassLabel).orderBy(rand()).limit(rPerClass);
/* Join R samples and broadcast rSamples to all nodes. */
rSamples ← rMinSamples.union(rMajSamples);
spark.broadcast(rSamples);
/* Transform feature space into dissimilarity space on train and test data by using the
   dissimilarityTransform procedure. */
trainDissimData ← trainData.map(row => dissimilarityTransform(row, fractionalDistance));
testDissimData ← testData.map(row => dissimilarityTransform(row, fractionalDistance));
/* Save the transformed data into files. */
trainDissimData.write(dissimilarityTrainFile);
testDissimData.write(dissimilarityTestFile);

Procedure dissimilarityTransform(dataSample, fractionalDistance):
    /* Initialize dissimSample and classLabel variables. */
    dissimSample ← null;
    classLabel ← dataSample.classLabel;
    /* Loop through all elements of rSamples. */
    for $i$ ← 0 to rSamples.size do
        /* Initilized sum variable. */
        sum ← 0;
        /* Loop through all features of dataSample. */
        for $j$ ← 0 to dataSample.size do
            /* Compute the distance between the dataSample and rSample. */
            sum += scala.math.pow((dataSample[j] - rSamples[i][j]).abs, fractionalDistance);
        distance ← scala.math.pow(sum, 1/fractionalDistance);
        /* Add the new sample in the dissimilarity space into dissimSample. */
        dissimSample.add(distance);
    /* Return the class label and the sample data in the dissimilarity space. */
    return (classLabel,dissimSample)
```

2.3. Knn-Is and SMOTE in Big Data

The kNN-IS model significantly improves the k-nearest neighbor rule for large datasets by optimizing various aspects of the computational process within Spark. This model not only addresses the scalability issues faced with large datasets but also improves the overall efficiency of the classification process.

It consists of two main steps that allow for it to have a scalable behavior [24]. The map operation calculates the distances between each test sample and all training samples. Each node independently computes these distances during this phase, creating a class-distance vector for each test sample. This parallel computation is essential for handling large datasets efficiently, as it allows for the processing load to be distributed across multiple nodes.

The reduce phase then aggregates these class-distance vectors to determine the k-nearest neighbors for each test sample. This phase involves sorting the class-distance vectors in ascending order of distance and selecting the k smallest distances to identify the nearest neighbors. The algorithm efficiently consolidates the results from the distributed computations by leveraging the reduce function, ensuring that the final k-NN classification is accurate and scalable for Big Data applications.

In this paper, the original SMOTE-BD code was modified to work with LIBSVM files. Algorithms 2–4 show the SMOTE-BD and kNN-IS implemented in Spark, where

- Algorithm 2—the main algorithm of SMOTE-BD. This algorithm begins by reading the files and filtering the data by class. It then identifies the k-nearest neighbors (kNNs) of the minority class and calculates the amount of synthetic data to generate. Subsequently, it calls the procedure in Algorithm 3 to create the synthetic data.
- Algorithm 3—Synthetic Data Generation. Based on the identified kNNs, this algorithm runs on the Spark data partition. For each data point in the partition, it selects a random nearest neighbor and calls Algorithm 4 to interpolate new data points from the two minority class points.
- Algorithm 4—Interpolation Process. This algorithm receives two data points and returns a new synthetic data point based on the distance between the two points, multiplied by a random value.

Algorithm 2: Adapted SMOTE-BD

```
/* Input and output files are in LIBSVM format. */
/* trainDataFile is the train data in the dissimilarity space. */
Data: trainDataFile
/* minClassLabel is the minority class label used to filter the minority class data in the
   algorithm. */
/* k is the number of nearest neighbors to search in the kNN algorithm. */
/* numIterations is the number of iterations kNN-IS will perform to find the k-nearest
   neighbors; default to 1. */
Input: minClassLabel, k, numIterations
/* oversampledTrainDataFile is the filename used to save the oversampled dataset. */
Output: oversampledTrainDataFile
Load KNN-IS jar into Spark;
trainData ← spark.read.libsvm(trainDataFile).repartition();
countSamples ← trainData.count();
/* Filter minority class samples. */
minSamples ← trainData.filter(trainData.classLabel == minClassLabel);
countMinoirty ← minSamples.count();
countMajority ← countSamples - countMinority;
/* Compute how many artificial samples SMOTE needs to create per real minority class sample. */
sRate ← (countMajority-countMinority)/countMinority;
/* Using KNN-IS calculate the k-nearest neighbors of the minority class data samples. */
neighbours ← KNNIS.setup(minSamples,k).calculateNeighbours();
/* Broadcast the neighbors variable to all nodes. */
spark.broadcast(neighbours);
/* Initialize synthSamples and tmpSamples variables. */
synthSamples ← null;
tmpSamples ← null;
for $i < numIterations$ do
    /* Create synthetic samples from data partitions. */
    tmpSamples ← minSamples.mapPartitionsWithIndex(synthData(idx, pData, neighbours, sRate, k));
    if $synthamples == null$ then
        synthSamples ← tmpSamples;
    else
        /* Add new synthetic samples to existing synthetic samples. */
        synthSamples ← synthSamples.union(tmpSamples);
/* Add synthetic data to training data. */
outputData ← trainData.union(synthSamples);
/* Save the oversampled data set to a file. */
Result: outputData.write(oversampledTrainDataFile)
```

Algorithm 3: Procedure to compute the k nearest neighbors for each minority class sample and synthetic data generation

```
/* This procedure is executed at the partition data level in Spark. */
/* partitionData is the partition data used to create new data samples. */
/* neighbours is the data neighbors previously calculated and broadcast to all the
   nodes. */
/* createionRate is the number of synthetic data samples to be created. */
/* k is the number of nearest neighbors. */
Input: partitionData, neighbours, creationRate, k
/* newSamples are the synthetic data samples created. */
Output: newSamples
1 Procedure synthData(partitionData, neighbours, creationRate, k):
      /* Initialize newSamples variable. */
2     newSamples ← null;
      /* Loop through all data samples of the partition. */
3     forall element of partitionData do
          /* Create ‘‘creationRate’’ synthetic samples. */
4         for count ← 0 to creationRate do
              /* sample1 is the current data sample in the ForAll loop. */
5             sample1 ← element;
              /* sample2 is a random neighbor. */
6             sample2 ← neighbours[Random.nextInteger(k)];
              /* Create a new synthetic sample interpolating sample1 and sample2. */
7             sample ← interpolate(sample1, sample2);
              /* Add a new synthetic sample to newSamples. */
8             newSamples.add(sample);
      /* Return the created synthetic samples. */
9     return newSamples
```

Algorithm 4: Procedure of Interpolation Process

```
/* This procedure creates new sample data based on two provided samples. */
/* sample1 and sample2 are existing data samples used to create a new synthetic data
   sample. */
Input: sample1, sample2
/* newSample is a new data sample created based on the input samples. */
Output: newSample
1 Procedure interpolate(sample1, sample2):
      /* Initialize newSample variable. */
2     newSample ← null;
      /* Obtain a random number between 0 and 1. */
3     random ← Random.nextDouble();
      /* Loop over all elements of the data sample. */
4     for i ← 0 to sample1.size do
          /* Calculate a delta for the new synthetic sample. */
5         delta ← (sample1[i] - sample2[i]) × random;
          /* Create a new sample by adding a randomly generated delta. */
6         newSample[i] ← sample2[i] + delta;
      /* Return the new synthetic data. */
7     return newSample
```

2.4. Proposed Technique

The novel preprocessing technique introduced in this article (see Figure 1) aims to address four types of concurrent complexities: (i) Big Data, (ii) high dimensionality, (iii) imbalanced classes, and (iv) class overlapping.

The hybrid technique proposed in this section is structured in two phases: (i) a transformation from the feature space to a dissimilarity space is performed, utilizing fractional distances as a measure of similarity (see Algorithm 1), and (ii) the transformed dataset is balanced using the SMOTE-BD [23], which uses the Euclidean distance. All experiments were run in the Spark distributed system.

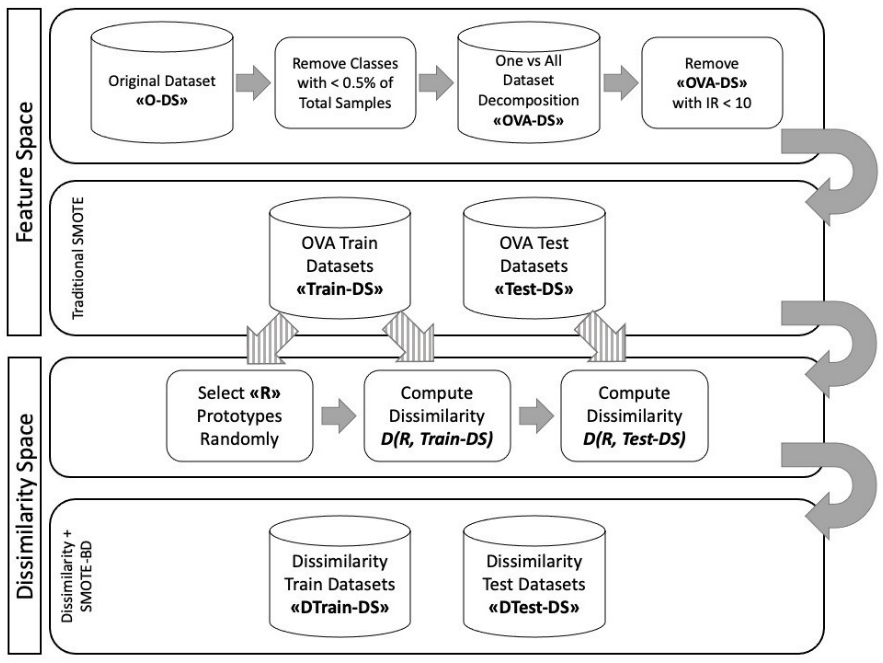
Feature Space
Original Dataset «O-DS»
Remove Classes with < 0.5% of Total Samples
One vs All Dataset Decomposition «OVA-DS»
Remove «OVA-DS» with IR < 10
Traditional SMOTE
OVA Train Datasets «Train-DS»
OVA Test Datasets «Test-DS»
Dissimilarity Space
Select «R» Prototypes Randomly
Compute Dissimilarity D(R, Train-DS)
Compute Dissimilarity D(R, Test-DS)
Dissimilarity + SMOTE-BD
Dissimilarity Train Datasets «DTrain-DS»
Dissimilarity Test Datasets «DTest-DS»

Figure 1. Proposed technique for preprocessing imbalanced high-dimensional Big Data datasets.

3. Experimental Setup

For the experiments presented in this paper, we created an artificial dataset from an original dataset that contains 21,025 instances, 24,832 features, and 17 numerically labeled classes from 0 to 16.

The original dataset, known as the Indian pine scene, was obtained from the Computational Intelligence Group site [40]. This dataset was collected by NASA's AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) sensor, which provides 224 bands and a spectral resolution of 10.0 nm [41]. It consists of one-third of the forest area, and the rest is the cultivation field or residual natural vegetation indices. Table 1 depicts the ground truth classes and their corresponding instances for the Indian pine scene.

Although datasets with high dimensionality are available in several repositories, some have characteristics that fall outside the article's scope, such as a small number of instances or sparse data. Inspired by the works of Rendón et al. [42] and Charte et al. [43], which demonstrated how a neural network can be used for transforming data in order to find other features, we employed a Multi-layer Perceptron (MLP) model that was trained using the full dataset. Following the parameter specification in [42], we built the MLP and extracted from the output of a hidden layer the newly transformed dataset with a high dimensionality of 24,832 features. Due to the dataset size, it will be available upon request.

Since we are interested in two-class imbalance datasets, we joined several classes to form the majority class, leaving a single class as the minority class. During this process, we omitted classes that constitute less than 0.5% of the dataset (i.e., Alfalfa, Grass-pasture-mowed, Oats, and Stone-Steel-Towers). This resulted in a total of 9 imbalanced datasets with several imbalance ratios (IR). Table 2 shows all the characteristics of the datasets, where the number that appears in the dataset name denotes the class chosen as the minority class.

All the datasets were partitioned using 70% of instances for training and the rest 30% for testing. We used the Decision Tree (DT) classifier from the Spark MLlib library. The DT classifier was chosen with the GINI impurity measure and configured with a maximum depth of 5 and 32 bins.

The fractional values of p used were 0.25, 0.33, 0.50, 0.66, and 0.75. This choice was based on considerations from published works. For the dissimilarity representation, the selected sizes for R included 10, 50, 100, 200, 500, 1000, and 2000 instances. In each selection, *R* was composed, ensuring an equitable representation of examples from both the majority and minority classes.

Table 1. Number of instances per class and percentage relative to the total samples for the Indian pine dataset.

Class	Numeric Class	Instances	Percentage
Background	0	10,776	51.25%
Alfalfa	1	46	0.22%
Corn-notill	2	1428	6.79%
Corn-mintill	3	830	3.95%
Corn	4	237	1.13%
Grass-pasture	5	483	2.30%
Grass-trees	6	730	3.47%
Grass-pasture-mowed	7	28	0.13%
Hay-windrowed	8	478	2.27%
Oats	9	20	0.10%
Soybean-notill	10	972	4.62%
Soybean-mintill	11	2455	11.68%
Soybean-clean	12	593	2.82%
Wheat	13	205	0.98%
Woods	14	1265	6.02%
Buildings-Grass-Trees-Drives	15	386	1.84%
Stone-Steel-Towers	16	93	0.44%
Total:	-	21,025	100%

Table 2. Characteristics of the imbalanced Big Data datasets (#Majority: number of instances in the majority class, #Minority: number of instances in the minority class, IR: imbalance ratio defined as the number of instances of the majority class divided by the number of instances of the minority class, Total: Total number of instances in the dataset).

Dataset	# Majority	# Minority	Total	IR
NewIndian2	19,410	1428	20,838	13.59
NewIndian3	20,008	830	20,838	24.11
NewIndian5	20,355	483	20,838	42.14
NewIndian6	20,108	730	20,838	27.55
NewIndian8	20,360	478	20,838	42.59
NewIndian10	19,866	972	20,838	20.44
NewIndian12	20,245	593	20,838	34.14
NewIndian14	19,573	1265	20,838	15.47
NewIndian15	20,452	386	20,838	52.98

The Big Data architecture was developed on Google Cloud using the credit grant obtained from the "Google for Education" program [44]. The Spark 3.1.1 cluster was configured with one master node equipped with 32 GB of memory and 8 vCPUs, and three slave nodes, each with 64 GB of memory and 16 vCPUs.

Performance Metrics

The DT classifier is evaluated using a confusion matrix, a square matrix where the entries (i, j) contain the number of correct and incorrect predictions [45]. Table 3 displays a 2×2 table for a two-class problem, where the columns represent the classifier's estimated output and the rows indicate the actual classes. The elements on the main diagonal represent the correct number of predictions for the positive and negative classes, while the other entries represent the errors.

Table 3. Confusion matrix for a two-class problem.

	Positive Prediction	**Negative Prediction**
Actual Positive	True Positives (TP)	False Negatives (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

Typically, the metric employed for the effectiveness of a classifier is the accuracy, denoted as

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

However, it has been demonstrated that this metric is unsuitable when the dataset is imbalanced; hence, other alternatives are used [45]. Examples of alternative metrics that evaluate the effectiveness on each class are

- **The True Positive Rate**, which represents the proportion of positive examples correctly classified.
$$TPR = \frac{TP}{TP + FN} \tag{5}$$
- **The True Negative Rate** or Recall, which represents the proportion of negative examples correctly classified.
$$TNR = \frac{TN}{TN + FP} \tag{6}$$
- **The False Negative Rate**, which is the proportion of positive examples misclassified.
$$FNR = \frac{FN}{TP + FN} \tag{7}$$
- **The False Positive Rate**, which is the proportion of negative examples misclassified.
$$FPR = \frac{FP}{TN + FP} \tag{8}$$

A metric encompassing individual rates per class is the geometric mean,

$$G - mean = \sqrt{TPR \times TNR}, \tag{9}$$

which maximizes the effectiveness of each class while maintaining balance.

The classifiers can also be evaluated using graphical methods, which allow for the visualization of their effectiveness [46]. One of the most popular strategies is the ROC graph (Receiver Operating Characteristic curve). In this two-dimensional space, curves represent the trade-off between TPR (Y-axis) and FPR (X-axis). A scalar value can be derived from the ROC space by calculating the area under the curve, known as the AUC (Area Under the ROC curve). In the literature, there are several ways of computing the AUC, here, we have adopted the version that captures a single point on the ROC curve, also called balanced accuracy, computed as [47]

$$AUC = \frac{TPR + TNR}{2}. \tag{10}$$

4. Results and Discussion

The experiments were conducted to evaluate three strategies for dealing with imbalanced and high-dimensional Big Data datasets: (1) SMOTE, (2) dissimilarity with fractional norms (D_{d_p}), and (3) the joint use of dissimilarity and SMOTE (D_{d_p}+ SMOTE). For clarity and simplicity, in strategies (2) and (3), we only selected the two best results in terms of AUC and G-mean.

Table 4 presents the experimental results for each strategy used on the nine datasets. It includes the values of p in the Minkowsky distance metric used in constructing the dissimilarity matrix and the resulting dimensionality (number of features) obtained when applying this mapping. Additionally, the classification results on the original imbalanced dataset and with SMOTE are provided for comparison purposes.

The experimental results using SMOTE align well with the general conclusions from numerous studies on standard problems, indicating the effectiveness of dealing with the class imbalance problem. Despite the high dimensionality of the data, the use of SMOTE consistently improves both the AUC and G-mean across all cases when compared with the baseline strategy (without the preprocessing technique). However, this enhancement comes at the cost of increasing the size of the dataset.

When the imbalanced dataset is transformed into a dissimilarity representation with fractional norms, the results generally exhibit lower performance than those obtained with the original dataset. However, it is important to note that the dataset continues presenting the class imbalance problem in this new representation. Despite this, the decrease in performance is not always drastic, even though the dimensionality is reduced by 92% and 98% in some cases. Similarly, while it is challenging to recommend a specific value of p, in most situations, better results are observed when $p < 1$.

Table 4. Experimental results. The best AUC and $G - mean$ results are highlighted in purple and grey for each dataset, respectively.

Name	Dataset	p	Features	TPR	TNR	AUC	$G - mean$
Imbalanced	NewIndian2	–	24,832	0.51402	0.97645	0.74524	0.70846
SMOTE		2.00	24,832	0.85047	0.86215	**0.85631**	**0.85629**
D_{d_p}		0.50	1000	0.38318	0.97800	0.68059	0.61217
		0.33	2000	0.35748	0.98247	0.66997	0.59263
D_{d_p}+SMOTE		1.00	500	0.91121	0.78206	0.84664	0.84417
		0.50	2000	0.89252	0.79477	0.84365	0.84223
Imbalanced	NewIndian3	–	24,832	0.45977	0.98814	0.72395	0.67403
SMOTE		2.00	24,832	0.75096	0.93049	0.84073	0.83592
D_{d_p}		0.25	50	0.36398	0.99114	0.67756	0.60063
		0.33	2000	0.32184	0.99566	0.65875	0.56608
D_{d_p}+SMOTE		0.25	100	0.93870	0.84378	**0.89124**	**0.88997**
		0.66	500	0.96169	0.81988	0.89078	0.88796
Imbalanced	NewIndian5	–	24,832	0.61268	0.99672	0.80470	0.78145
SMOTE		2.00	24,832	0.80282	0.95937	0.88109	0.87761
D_{d_p}		0.33	50	0.63380	0.99394	0.81387	0.79370
		0.75	500	0.62676	0.99492	0.81084	0.78967
D_{d_p}+SMOTE		0.25	2000	0.90141	0.93381	**0.91761**	**0.91747**
		0.25	500	0.88732	0.94725	0.91729	0.91680
Imbalanced	NewIndian6	–	24,832	0.57078	0.99121	0.78099	0.75217
SMOTE		2.00	24,832	0.86758	0.96665	0.91711	0.91578
D_{d_p}		0.75	1000	0.58447	0.98722	0.78585	0.75961
		0.25	2000	0.57078	0.99121	0.78099	0.75217
D_{d_p}+SMOTE		0.66	1000	0.96804	0.89132	**0.92968**	**0.92889**
		0.75	2000	0.95434	0.89182	0.92308	0.92255
Imbalanced	NewIndian8	–	24,832	0.92373	0.99461	0.95917	0.95852
SMOTE		2.00	24,832	0.94915	0.99249	0.97082	0.97058
D_{d_p}		0.33	500	0.95763	0.99510	0.97637	0.97619
		0.25	500	0.94915	0.99543	0.97229	0.97202
D_{d_p} + SMOTE		1.00	50	1.00000	0.98890	**0.99445**	**0.99444**
		0.50	500	0.99153	0.99184	0.99168	0.99168
Imbalanced	NewIndian10	–	24,832	0.59375	0.98456	0.78915	0.76458
SMOTE		2.00	24,832	0.86111	0.91658	0.88885	0.88841
D_{d_p}		2.00	100	0.58333	0.97583	0.77958	0.75448
		2.00	1000	0.58333	0.97012	0.77673	0.75227
D_{d_p}+SMOTE		2.00	2000	0.93056	0.89225	**0.91140**	**0.91120**
		0.66	500	0.91319	0.90500	0.90910	0.90909

Table 4. *Cont.*

Name	Dataset	p	Features	TPR	TNR	AUC	$G - mean$
Imbalanced	NewIndian12	–	24,832	0.44767	0.99177	0.71972	0.66633
SMOTE		2.00	24,832	0.78488	0.91916	0.85202	0.84937
D_{d_p}		0.33	2000	0.31395	0.99325	0.65360	0.55842
		0.33	1000	0.30233	0.99341	0.64787	0.54803
D_{d_p} + SMOTE		0.50	500	0.96512	0.76934	**0.86723**	**0.86169**
		0.25	1000	0.91279	0.81017	0.86148	0.85995
Imbalanced	NewIndian14	–	24,832	0.54595	0.96886	0.75740	0.72728
SMOTE		2.00	24,832	0.85676	0.91797	0.88736	0.88684
D_{d_p}		0.25	1000	0.36486	0.98434	0.67460	0.59929
		0.25	100	0.37027	0.97294	0.67161	0.60021
D_{d_p} + SMOTE		1.00	10	0.97838	0.85160	**0.91499**	**0.91279**
		2.00	1000	0.96486	0.86266	0.91376	0.91233
Imbalanced	NewIndian15	–	24 ,832	0.06838	0.99445	0.53141	0.26076
SMOTE		2.00	24,832	0.67521	0.84989	0.76255	0.75754
D_{d_p}		0.50	2000	0.02564	0.99853	0.51209	0.16001
		0.50	1000	0.02564	0.99788	0.51176	0.15996
D_{d_p} + SMOTE		0.33	2000	0.91453	0.71888	**0.81670**	**0.81082**
		0.50	10	0.94872	0.68364	0.81618	0.80534

The proposed approach of combining dissimilarity with fractional norms followed by SMOTE (D_{d_p}+SMOTE) demonstrated the most promising performance in terms of both AUC and G-mean. Notably, this improvement was achieved while maintaining a lower dimensionality compared to the original datasets. It demonstrates that integrating these techniques with $p < 1$ can synergistically enhance the model's performance on the minority class.

In order to determine whether there are significant differences in the results, we used a nonparametric statistical test called the Friedman test. Additionally, a post-doc test was conducted using Bonferrini–Dunn's method to determine which algorithms perform better, equally, or worse. Table 5 summarizes the average ranks of each strategy based on the Friedman test, where the strategy with the best performance obtains the lowest rank, and the worst-performing approach receives the highest rank [48].

According to the results, D_{d_p}+SMOTE ranked the best overall, with an average rank of 1.1, indicating that it consistently outperformed the other strategies across the different datasets. SMOTE was ranked second with an average rank of 2. The imbalanced strategy ranked third, and finally, D_{d_p} obtained the highest average rank, indicating that it was the worst-performing strategy. Based on the Iman and Davenport statistic F_F, distributed according to the F distribution with four algorithms and nine datasets, yielding a value of 31.512, and a computed p-value of 0.000000017243, we reject the null hypothesis at the significance level of $\alpha = 0.05$. This indicates that there are significant differences among the algorithms tested.

Table 5. Average rankings of the algorithms (Friedman) using *AUC*.

Algorithm	Ranking (AUC)
Imbalanced	3.3
SMOTE	2
D_{d_p}	3.6
D_{d_p}+SMOTE	1.1

Further analysis using the Bonferroni–Dunn's procedure as a post hoc method is presented in Table 6, showing the p-values obtained over the results of the Friedman test with $\alpha = 0.05$, using D_{d_p}+SMOTE as the control algorithm. With a p-value of 0.016667, we reject the null hypothesis for the following pairs, indicating that the control algorithm is superior: D_{d_p}+SMOTE vs. Ddp and D_{d_p}+SMOTE vs. Imbalanced. However, there is no significant difference between D_{d_p}+SMOTE vs. SMOTE, suggesting that they perform similarly well.

Table 6. Post hoc comparison with $\alpha = 0.05$ and D_{d_p}+SMOTE as algorithm control.

i	**Algorithm**	$z = (R_0 - R_i)/SE$	p
3	D_{d_p}	4.016632	0.000059
2	Imbalanced	3.651484	0.000261
1	SMOTE	1.460593	0.144127

5. Conclusions

The motivation for this study lies in addressing significant challenges encountered in Big Data analytics, specifically class imbalance, high dimensionality, and class overlapping. In this paper, we propose a hybrid approach based on several data-level solutions for addressing the class imbalance problem in Big Data datasets with high dimensionality. The proposal combines a dissimilarity approach to transform the datasets into a low-dimensional space using fractional norms and the well-known SMOTE technique. This strategy can improve the quality of the data by reducing the dimensionality and balancing the classes. Our results show that the preprocessed dataset achieved better or comparable AUC and G-mean results compared to those obtained from the original feature space, despite having significantly fewer dimensions—often less than 10% of the original. This substantial reduction in dimensionality enhances the efficiency of other preprocessing techniques and improves overall computational performance and resource utilization. This study utilized pre-selected fractional distances and a specified number of randomly selected instances for R. Determining the optimal fractional distance and the appropriate number of instances for transforming the feature space into the dissimilarity space remains largely dataset-dependent.

Despite the successful results, this study has several limitations: (i) a comparative study with other techniques proposed in the literature, (ii) the datasets used in this research were limited to nine specific datasets, (iii) the number of attributes in the datasets, and (iv) the use of a single classifier.

Future research should aim to optimize prototype selection for the dissimilarity transformation further and explore various fractional distances to assess the robustness and adaptability of the proposed method across a broader range of datasets and classification challenges.

Author Contributions: Conceptualization, A.B., V.G., R.A., and R.F.-J.; methodology, J.S.S.; software, A.B.; validation, A.B., V.G., and J.S.S.; formal analysis, A.B., V.G., R.A., and R.F.-J.; investigation, A.B.; resources, A.B.; data curation, A.B. and R.A.; writing—original draft preparation, A.B.; writing—review and editing, V.G., R.A., R.F.-J., and J.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received Google Cloud credits from the Google for Education program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the second author, V.G., upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Domo, I. Data Never Sleeps 11.0. Available online: https://www.domo.com/learn/infographic/data-never-sleeps-11 (accessed on 10 May 2024).
2. Reinsel, D.; Gantz, J.; Rydning, J. Data Age 2025: The Evolution of Data to Life-Critical. In *Don't Focus on Big Data*; Focus on the Data That's Big; Technical Report; SEAGATE: Dublin, Ireland, 2017.
3. Ducange, P.; Fazzolari, M.; Marcelloni, F. An overview of recent distributed algorithms for learning fuzzy models in Big Data classification. *J. Big Data* **2020**, *7*, 19. [CrossRef]
4. Triguero, I.; Galar, M. *Large-Scale Data Analytics with Python and Spark*; Cambridge University Press: Cambridge, UK, 2024.
5. Anjum, M.; Min, H.; Ahmed, Z. Trivial State Fuzzy Processing for Error Reduction in Healthcare Big Data Analysis towards Precision Diagnosis. *Bioengineering* **2024**, *11*, 539. [CrossRef] [PubMed]

6. Onyejekwe, E.R.; Sherifi, D.; Ching, H. Perspectives on Big Data and Big Data Analytics in Healthcare. *Perspect. Health Inf. Manag.* **2024**, *21*, 43.
7. Zhou, L.; Pan, S.; Wang, J.; Vasilakos, A.V. Machine learning on big data: Opportunities and challenges. *Neurocomputing* **2017**, *237*, 350–361. [CrossRef]
8. Gupta, P.; Sharma, A.; Jindal, R. Scalable machine-learning algorithms for big data analytics: A comprehensive review. *WIREs Data Min. Knowl. Discov.* **2016**, *6*, 194–214. [CrossRef]
9. Tosi, D.; Kokaj, R.; Roccetti, M. 15 years of Big Data: A systematic literature review. *J. Big Data* **2024**, *11*, 73. [CrossRef]
10. Ali El-Sayed Ali, H.; Alham, M.H.; Ibrahim, D.K. Big data resolving using Apache Spark for load forecasting and demand response in smart grid: A case study of Low Carbon London Project. *J. Big Data* **2024**, *11*, 59. [CrossRef]
11. Ngiam, K.Y.; Khor, I.W. Big data and machine learning algorithms for health-care delivery. *Lancet. Oncol.* **2019**, *20*, e262–e273. [CrossRef] [PubMed]
12. Ng, A. AI Doesn't Have to Be Too Complicated or Expensive for Your Business. Harvard Business Review, 2021. Available online: https://hbr.org/2021/07/ai-doesnt-have-to-be-too-complicated-or-expensive-for-your-business (accessed on 2 December 2023).
13. Sambasivan, N.; Kapania, S.; Highfill, H.; Akrong, D.; Paritosh, P.; Aroyo, L.M. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; Association for Computing Machinery: New York, NY, USA, 2021. [CrossRef]
14. Pagano, T.P.; Loureiro, R.B.; Lisboa, F.V.N.; Peixoto, R.M.; Guimarães, G.A.S.; Cruz, G.O.R.; Araujo, M.M.; Santos, L.L.; Cruz, M.A.S.; Oliveira, E.L.S.; et al. Bias and Unfairness in Machine Learning Models: A Systematic Review on Datasets, Tools, Fairness Metrics, and Identification and Mitigation Methods. *Big Data Cogn. Comput.* **2023**, *7*, 15. [CrossRef]
15. Kumar, A.; Singh, D.; Shankar Yadav, R. Class overlap handling methods in imbalanced domain: A comprehensive survey. *Multimed. Tools Appl.* **2024.** [CrossRef]
16. Hasanin, T.; Khoshgoftaar, T.M.; Leevy, J.L.; Bauder, R.A. Investigating class rarity in big data. *J. Big Data* **2020**, *7*, 23. [CrossRef]
17. Triguero, I.; García-Gil, D.; Maillo, J.; Luengo, J.; García, S.; Herrera, F. Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. *WIREs Data Min. Knowl. Discov.* **2019**, *9*, e1289. [CrossRef]
18. Santos, M.S.; Abreu, P.H.; Japkowicz, N.; Fernández, A.; Santos, J. A unifying view of class overlap and imbalance: Key concepts, multi-view panorama, and open avenues for research. *Inf. Fusion* **2023**, *89*, 228–253. [CrossRef]
19. Maldonado, S.; López, J. Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for SVM classification. *Appl. Soft Comput.* **2018**, *67*, 94–105. [CrossRef]
20. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Int. Res.* **2002**, *16*, 321–357. [CrossRef]
21. Kovács, G. SMOTE-variants: A python implementation of 85 minority oversampling techniques. *Neurocomputing* **2019**, *366*, 352–354. [CrossRef]
22. Sisodia, D.; Sisodia, D.S. Data Sampling Strategies for Click Fraud Detection Using Imbalanced User Click Data of Online Advertising: An Empirical Review. *IETE Tech. Rev.* **2022**, *39*, 789–798. [CrossRef]
23. Basgall, M.J.; Hasperué, W.; Naiouf, M.; Fernández, A.; Herrera, F. SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data. *J. Comput. Sci. Technol.* **2018**, *18*, e23. [CrossRef]
24. Maillo, J.; Ramírez, S.; Triguero, I.; Herrera, F. kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data. *Knowl.-Based Syst.* **2017**, *117*, 3–15. [CrossRef]
25. Elreedy, D.; Atiya, A.F. A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance. *Inf. Sci.* **2019**, *505*, 32–64. [CrossRef]
26. Maldonado, S.; López, J.; Vairetti, C. An alternative SMOTE oversampling strategy for high-dimensional datasets. *Appl. Soft Comput.* **2019**, *76*, 380–389. [CrossRef]
27. Maldonado, S.; Vairetti, C.; Fernandez, A.; Herrera, F. FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification. *Pattern Recognit.* **2022**, *124*, 108511. [CrossRef]
28. Pękalska, E.; Duin, R.P. Dissimilarity representations allow for building good classifiers. *Pattern Recognit. Lett.* **2002**, *23*, 943–956. [CrossRef]
29. Costa, Y.M.G.; Bertolini, D.; Britto, A.S.; Cavalcanti, G.D.C.; Oliveira, L.E.S. The dissimilarity approach: A review. *Artif. Intell. Rev.* **2020**, *53*, 2783–2808. [CrossRef]
30. Thudumu, S.; Branch, P.; Jin, J.; Singh, J.J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **2020**, *7*, 42. [CrossRef]
31. Flexer, A.; Schnitzer, D. Choosing l-norms in high-dimensional spaces based on hub analysis. *Neurocomputing* **2015**, *169*, 281–287. [CrossRef] [PubMed]
32. Tomasev, N.; Radovanovic, M.; Mladenic, D.; Ivanovic, M. The Role of Hubness in Clustering High-Dimensional Data. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 739–751. [CrossRef]
33. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 420–434. [CrossRef]
34. Mirkes, E.M.; Allohibi, J.; Gorban, A. Fractional Norms and Quasinorms Do Not Help to Overcome the Curse of Dimensionality. *Entropy* **2020**, *22*, 1105. [CrossRef] [PubMed]

35. Cormode, G.; Indyk, P.; Koudas, N.; Muthukrishnan, S. Fast mining of massive tabular data via approximate distance computations. In Proceedings of the 18th International Conference on Data Engineering, ICDE-02, San Jose, CA, USA, 26 February–1 March 2002. [CrossRef]
36. Gorban, A.N.; Mirkes, E.M.; Zinovyev, A. Data analysis with arbitrary error measures approximated by piece-wise quadratic PQSQ functions. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
37. Duin, R.P.; Pękalska, E. *The Dissimilarity Representation for Non-Euclidean Pattern Recognition, a Tutorial*; Technical Report; Delf University of Technology: Delft, The Netherlands, 2011.
38. Duin, R.P.; Pękalska, E. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognit. Lett.* **2012**, *33*, 826–832. [CrossRef]
39. García, V.; Marqués, A.I.; Sánchez, J.S.; Ochoa-Domínguez, H.J. Dissimilarity-Based Linear Models for Corporate Bankruptcy Prediction. *Comput. Econ.* **2019**, *53*, 1019–1031. [CrossRef]
40. Graña, M.; Veganzons, M.; B, A. Indian Pines Dataset. Available online: https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Indian_Pines (accessed on 2 December 2023).
41. NASA. AVIRIS: Airborne Visible - Infrared Imaging Spectrometer. Available online: https://aviris.jpl.nasa.gov/data/index.html (accessed on 2 December 2023).
42. Rendón, E.; Alejo, R.; Castorena, C.; Isidro-Ortega, F.J.; Granda-Gutiérrez, E.E. Data Sampling Methods to Deal With the Big Data Multi-Class Imbalance Problem. *Appl. Sci.* **2020**, *10*, 1276. [CrossRef]
43. Charte, D.; Charte, F.; Herrera, F. Reducing Data Complexity Using Autoencoders With Class-Informed Loss Functions. *Pattern Anal. Mach. Intell.* **2022**, *44*, 9549–9560. [CrossRef] [PubMed]
44. Google. Programas de educación superior de Google Cloud. Available online: https://cloud.google.com/edu/ (accessed on 7 July 2022).
45. Japkowicz, N. *Evaluating Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2011. [CrossRef]
46. Prati, R.C.; Batista, G.E.A.P.A.; Monard, M.C. A Survey on Graphical Methods for Classification Predictive Performance Evaluation. *IEEE Trans. Knowl. Data Eng.* **2011**, *23*, 1601–1618. [CrossRef]
47. Branco, P.; Torgo, L.; Ribeiro, R.P. A Survey of Predictive Modeling on Imbalanced Domains. *ACM Comput. Surv.* **2016**, *49*, 1–50. [CrossRef]
48. García, S.; Fernández, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.* **2010**, *180*, 2044–2064. [CrossRef]