

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

**Unsupervised Dimensionality Reduction in Big Data via
Massive Parallel Processing with MapReduce and Resilient
Distributed Datasets**

Jadson José Monteiro Oliveira

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Jadson José Monteiro Oliveira

Unsupervised Dimensionality Reduction in Big Data via Massive Parallel Processing with MapReduce and Resilient Distributed Datasets

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
January 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

048u Oliveira, Jadson Jose Monteiro
Unsupervised Dimensionality Reduction in Big
Data via Massive Parallel Processing with MapReduce
and Resilient Distributed Datasets / Jadson Jose
Monteiro Oliveira; orientador Robson Leonardo
Ferreira Cordeiro. -- São Carlos, 2021.
84 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2021.

1. Unsupervised Dimensionality Reduction. 2.
Descriptive Data Mining. 3. Big Data. 4. Fractal
Theory. I. Cordeiro, Robson Leonardo Ferreira,
orient. II. Título.

Jadson José Monteiro Oliveira

Redução de Dimensionalidade Não-Supervisionada em *Big Data* utilizando Processamento Paralelo com *MapReduce* e *Resilient Distributed Datasets*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
Janeiro de 2021

*This dissertation is dedicated to everyone who helped me get here
and to those who will benefit from these results.*

ACKNOWLEDGEMENTS

My thanks go out to all those who, in a way, contributed to the development of this work. To my whole family, especially my mother, grandmother, siblings, and stepfather, who always motivated me to move on and believe in the realization of my dreams and goals. They were the basis for all of this to become possible, where even in difficulties, they helped me a lot.

To my advisor Professor Robson, for all the support, presence, and dedication. For guiding me in all stages of this work, for the hours and hours of reviews, and for not measuring efforts to contribute with our results. He provided me a great growth in all aspects of my life.

For all GBDI friends and professors, for always sharing life experiences and scientific knowledge. Special thanks to friends Eugênio Cabral, Guilherme Zobot, Jessica Andressa, Leonardo Mauro, Lucas Kunze, Lucas Scabora, Mirela Cazzolato, and Thabata Amaral. They were always willing to help and contribute something and were exceptional for my development.

I thank the ICMC-USP for all the physical and organizational structure provided. To all employees who also contributed daily to the well-being of everyone at the institution.

Finally, I would like to thank CNPq [grant 166887 / 2017-0], CAPES [grant 001], FAPESP [grants 2018 / 05714-5 and 2016 / 17078-0], Microsoft Azure Research, and Amazon Web Services Cloud Credits for Research.

"The journey is what brings us happiness not the destination."

(Dan Millman)

RESUMO

OLIVEIRA, J. J. M. **Redução de Dimensionalidade Não-Supervisionada em *Big Data* utilizando Processamento Paralelo com *MapReduce* e *Resilient Distributed Datasets***. 2021. 81 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

O volume e a complexidade dos dados gerados em aplicações científicas e comerciais vêm crescendo exponencialmente em diversas áreas. Hoje, é comum a necessidade de encontrar padrões em *Terabytes* ou até mesmo em *Petabytes* de dados complexos, como em coleções de imagens, medições climáticas, impressões digitais e grandes grafos extraídos da Web ou de Redes Sociais. Por exemplo, como analisar *Terabytes* de dados oriundos de décadas de medições climáticas frequentes, compostos por dezenas de atributos climáticos como temperaturas, precipitação de chuva e umidade do ar, a fim de identificar padrões que antecedam eventos climáticos extremos para uso em sistemas de alerta? Um fato bem conhecido em análise de dados complexos é que a busca por padrões requer pré-processamento por redução de dimensionalidade, devido a um problema conhecido como “maldição da alta dimensionalidade”. Hoje, poucos trabalhos permitem reduzir, de forma eficaz, a dimensionalidade de tais dados em escala de *Terabytes* e *Petabytes* – referenciados nesta monografia como *Big Data* – visto que é extremamente desejável processamento paralelo em massa, escalabilidade linear em relação ao número de objetos, e capacidade para detectar os mais diversos tipos de correlações entre os atributos do conjunto de dados. Este trabalho de mestrado apresenta um estudo aprofundado, comparando duas abordagens distintas para redução de dimensionalidade em *Big Data*: (a) uma abordagem padrão, baseada na preservação da variância dos dados, e; (b) uma alternativa, baseada na Teoria de Fractais, que é raramente explorada na literatura. Para esta última nós propomos um algoritmo rápido e escalável baseado no modelo *MapReduce* e na estrutura de *Resilient Distributed Datasets*, utilizando uma nova estratégia de particionamento no conjunto de atributos que nos habilita a processar dados de alta dimensionalidade. Ambas as estratégias foram avaliadas a partir da inserção de atributos redundantes formados por correlações de diversos tipos, tais como linear, quadrática, logarítmica e exponencial, em 11 conjuntos de dados reais, e verificando a habilidade dessas abordagens em detectar tais redundâncias. Os resultados indicam que, pelo menos para grandes conjuntos de dados com dimensionalidade de até ~ 1.000 atributos, nossa técnica baseada em fractais é a melhor opção, visto que ela removeu com alta precisão os atributos redundantes em quase todos os casos, ao contrário das abordagens baseadas em variância, mesmo quando utilizada a técnica KPCA que é feita para detectar correlações não lineares.

Palavras-chave: Redução de Dimensionalidade Não-Supervisionada, Mineração de Dados Descritiva, *Big Data*, Teoria de Fractais.

ABSTRACT

OLIVEIRA, J. J. M. **Unsupervised Dimensionality Reduction in Big Data via Massive Parallel Processing with MapReduce and Resilient Distributed Datasets**. 2021. 81 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

The volume and complexity of data generated in scientific and commercial applications have been growing exponentially in many areas. Nowadays, it is common the need for finding patterns in Terabytes or even Petabytes of complex data, such as image collections, climate measurements, fingerprints and large graphs extracted from the Web or from Social Networks. For example, how to analyze Terabytes of data from decades of frequent climate measurements comprised of dozens of climatic features, such as temperatures, rainfall and air humidity, so to identify patterns that precede extreme weather events for use in alert systems? A well-known fact in complex data analysis is that the search for patterns requires preprocessing by means of dimensionality reduction, due to a problem known as the “curse of high-dimensionality”. Nowadays, few techniques have been able to effectively reduce the dimensionality of such data in the scale of Terabytes or even Petabytes, which are referred to in this monograph as Big Data. In this context, massively parallel processing, linear scalability to the number of objects, and the ability to detect the most diverse types of correlations among the attributes are exceptionally desirable. This MSc work presents an in-depth study comparing two distinct approaches for dimensionality reduction in Big Data: (a) a standard approach based on data variance preservation, and; (b) an alternative, Fractal-based solution that is rarely explored, for which we propose a fast and scalable algorithm based on MapReduce and concepts from Resilient Distributed Datasets, using a new attribute-set-partitioning strategy that enables us to process datasets of high dimensionality. We evaluated both strategies by inserting into 11 real-world datasets, redundant attributes formed by correlations of various types, such as linear, quadratic, logarithmic and exponential, and verifying the ability of these approaches to detect such redundancies. The results indicate that, at least for large datasets with up to $\sim 1,000$ attributes, our fractal-based technique is the best option. It removed redundant attributes in nearly all cases with high precision, as opposed to the standard variance-preservation approaches that presented considerably worse results even when applying the KPCA technique that is made to detect nonlinear correlations.

Keywords: Unsupervised Dimensionality Reduction, Descriptive Data Mining, Big Data, Fractal Theory.

LIST OF FIGURES

Figure 1 – Process of Knowledge Discovery in Databases.	30
Figure 2 – Points over a line with their embedded and intrinsic dimensionalities.	32
Figure 3 – Example of variance-based feature extraction: 3-dimensional points are projected into a 2-dimensional, newly transformed feature space that maximizes variance.	33
Figure 4 – Synthetic and real-world fractals with exact or statistical self-similarity.	35
Figure 5 – Ten points in a two-dimensional dataset divided by hyper-grids, and the corresponding hyper-quad-tree-like structure with two resolution levels.	35
Figure 6 – Traditional AutoEncoder representation.	36
Figure 7 – Serial Processing <i>versus</i> Distributed Processing.	38
Figure 8 – Counting words example in MapReduce model.	39
Figure 9 – Lineage graph example: Log files filtering.	41
Figure 10 – General pipeline of CurlRemover.	46
Figure 11 – Summary of algorithms.	47
Figure 12 – Example feature set partitioning with embedded dimensionality $E = 8$, intrinsic dimensionality $D = 3$, cardinality n and block size $k = 4$	53
Figure 13 – The cumulative explained variance of PCA, SVD and KPCA techniques. (a) and (b) respectively show the values obtained by PCA/SVD when applied on Susy original dataset, and Susy dataset with 500 additional redundant attributes formed by mixed correlations; (c) and (d) show the values obtained by KPCA under the same datasets. Note: similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.	64
Figure 14 – Experimental results of PCA, SVD, PUFS, KPCA and FReE when increasing the number of redundant attributes inserted in dataset Susy. Note: similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.	66

Figure 15 – Experimental results of PCA, SVD, PUFS, KPCA and FReE when increasing the number of attributes per correlation in dataset Susy. Note: similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.	67
Figure 16 – Experimental results on increasing the number of redundant attributes for other 10 real-world datasets from different domains. Note: these results are presented in a summarized form for brevity, since they are similar to those obtained from dataset Susy that were already detailed. The full results regarding these 10 datasets are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.	69
Figure 17 – Experimental results on increasing the number of original attributes per correlation for other 10 real-world datasets from different domains. Note: these results are presented in a summarized form for brevity, as they are similar to those obtained from dataset Susy that were already detailed. The full results for these 10 real datasets are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.	70
Figure 18 – Runtime of PCA, SVD, PUFS, KPCA and FReE in seconds when selecting features from random samples of Susy and FMA (highest dimensionality), up to the full datasets. Our proposed FReE scales linearly on the data size. . . .	71

LIST OF ALGORITHMS

Algorithm 1 – Fractal Redundancy Elimination (FReE)	54
Algorithm 2 – <code>remove_redundancy()</code>	54

LIST OF TABLES

Table 1 – Summary of datasets used to evaluate the techniques studied.	60
Table 2 – Equations used to generate redundant attributes.	60

LIST OF ABBREVIATIONS AND ACRONYMS

CSSP	Column Subset Selection Problem
GFS	Google File System
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
KDD	Knowledge Discovery in Databases
KPCA	Kernel Principal Component Analysis
MPI	Message Passing Interface
PCA	Principal Component Analysis
PPCA	Probabilistic Principal Component Analysis
PUFS	Parallel Unsupervised Feature Selection
RDD	Resilient Distributed Dataset
sPCA	Scalable Principal Component Analysis
SSVD	Stochastic Singular Value Decomposition
SVD	Singular Value Decomposition

CONTENTS

1	INTRODUCTION	25
1.1	Problem Definition, Hypothesis and Main Objectives	25
1.2	Main Contributions	27
1.3	Final Considerations	27
2	BACKGROUND CONCEPTS	29
2.1	Knowledge Discovery in Databases	29
2.2	Dimensionality Reduction in Very Large Datasets	31
2.2.1	<i>Dimensionality Reduction based on Data Variance</i>	33
2.2.2	<i>Fractal-based Dimensionality Reduction</i>	34
2.2.3	<i>Other Dimensionality Reduction Approaches</i>	36
2.3	Distributed Processing	37
2.3.1	<i>MapReduce Programming Model</i>	38
2.3.2	<i>Resilient Distributed Datasets</i>	40
2.4	Final Considerations	41
3	RELATED WORKS	43
3.1	Unsupervised Dimensionality Reduction	43
3.1.1	<i>Approaches based on Data Variance</i>	43
3.1.2	<i>Approaches based on the Fractal Theory</i>	45
3.2	Comparative Studies	48
3.3	Final Considerations	48
4	PROPOSED METHOD	51
4.1	Proposed Method	51
4.1.1	<i>Dimensionality Limitation</i>	51
4.1.2	<i>Cardinality Limitation</i>	53
4.1.3	<i>Computational Complexity Analysis</i>	55
4.1.4	<i>Final Considerations</i>	57
5	PROPOSED EVALUATION	59
5.1	Proposed Evaluation	59
5.1.1	<i>Evaluation</i>	59
5.1.2	<i>Algorithm settings</i>	61

5.1.3	<i>Final Considerations</i>	62
6	EXPERIMENTAL RESULTS	63
6.1	Experimental results	63
6.1.1	<i>Increasing the number of redundant attributes</i>	63
6.1.2	<i>Increasing the number of attributes per correlation</i>	66
6.1.3	<i>Summary of results from the other 10 real datasets</i>	68
6.1.4	<i>Scalability comparison between the approaches studied</i>	69
6.1.5	<i>Final Considerations</i>	71
7	CONCLUSION	73
7.1	Main Contributions of this MSc Work	73
7.2	Discussion and Future Work	74
	BIBLIOGRAPHY	77

INTRODUCTION

The volume and the complexity of data generated in scientific and commercial applications have been increasing in multiple domains, such as Biology, Physics, Medicine, Astronomy, among others. A study presented by [Dobre and Xhafa \(2014\)](#) reports that in 2014, the world was producing around 2.5 quintillions of bytes growing data every day. Additionally, [Gantz and Reinsel \(2012\)](#) states that by the year 2020, more than 40 Zettabytes of data will be generated and collected. This panorama has motivated the development of techniques that can automatically help users to analyze, understand and extract knowledge from large datasets, especially from complex data, such as collections of images, audio, data streams, social network graphs, DNA sequences and many others ([CORDEIRO; FALOUTSOS; TRAINA JR, 2013](#)).

1.1 Problem Definition, Hypothesis and Main Objectives

Although many real applications depend on the analysis of large datasets – for example, to process billions of images from Flickr¹ or Facebook² aimed at the support of targeted marketing – the best current algorithms tend to be inefficient or ineffective in data with many attributes ([Sun; Li, 2014](#)). The main challenge in analyzing data with many attributes is a phenomenon known as the “curse of high dimensionality”, which states that increasing the number of attributes in data objects leads to fast degradation in the performance and accuracy of many analytical algorithms ([GOLAY; KANEVSKI, 2017; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016](#)). The preprocessing by dimensionality reduction is the primary technique applied in these cases. It aims to decrease the number of attributes, thus reducing the effects of the high dimensionality and also the amount of data to be analyzed and stored, which is feasible because real-world data usually present non-uniform distributions and attribute correlations ([TUNG; XU; OOI, 2005; FALOUTSOS; KAMEL, 1994](#)). The existing methods are: supervised or unsupervised.

¹ <<https://www.flickr.com>>

² <<https://www.facebook.com>>

Supervised dimensionality reduction uses external knowledge, *e.g.*, label information, and aims at getting a subset of non-redundant attributes that are relevant to a very specific mining task, such as classification or regression (GOLAY; KANEVSKI, 2017). Unsupervised methods, by contrast, do not make use of a priori knowledge about the output and aim at removing all the redundant attributes. They can be used as preprocessing tools in a wide variety of descriptive data mining and machine learning tasks, such as clustering and outlier detection (GOLAY; KANEVSKI, 2017; CHENG; LI; LIU, 2017). The state-of-the-art methods often aim at preserving most of the data variance, using well-known strategies such as Principal Component Analysis (PCA), Kernel PCA (KPCA) and Singular Value Decomposition (SVD). Unfortunately, these techniques present a central drawback: they are either unable to identify and eliminate non-linear attribute correlations – PCA-based and SVD-based approaches – or they cannot process data of high cardinality with a reasonable computational cost – KPCA-based ones. Since correlations of these types are very likely to exist in real data – for example, in Biology, it is known that the co-expression patterns of genes in a gene network can be non-linear; in Physics, the pressure, volume and temperature of one ideal gas exhibit non-linear relationships (TUNG; XU; OOI, 2005) – and the amount of data has been increasing, this drawback compromises the usability of unsupervised dimensionality reduction as a whole. Thus, the central question to be answered in this work is: *how to effectively reduce the dimensionality of very large collections of complex data – hereafter referred to as Big Data – using massively parallel processing on large clusters of computers, by detecting both linear and non-linear correlations among the attributes?*

It is noteworthy that, in a previous work guided by the same advisor of this work, concepts from the Fractal Theory and massively parallel processing were applied in the development of a novel dimensionality reduction algorithm. The algorithm was able to obtain superior accuracy of results when compared to the other existing techniques studied until that moment. However, that algorithm has limited performance concerning computational processing efficiency. We identified that the main bottleneck of the algorithm is in disk writing and subsequent reading of large temporary files, performing in the worst case scenario, several complete readings of the analyzed dataset. Such a fact leads to the central hypothesis of this MSc work:

Hypothesis: the application of concepts from the Fractal Theory in dimensionality reduction tasks, using massively parallel processing via Apache Spark, and maximizing the use of main memory instead of secondary memory allows the development of an effective and efficient technique that is capable of reducing the dimensionality of Terabytes or even Petabytes of complex data.

To validate this hypothesis, we conducted an in-depth exploratory study comparing two distinct unsupervised dimensionality reduction approaches: (a) a well-known approach in the literature, based on data variance preservation, and; (b) an approach based on concepts from the Fractal Theory that is rarely explored in the literature to dimensionality reduction tasks. We also

developed a fast and scalable dimensionality reduction algorithm using Fractal Theory concepts and massively parallel processing with Apache Spark to make it feasible the analysis of very large datasets of high dimensionality.

1.2 Main Contributions

In summary, the main contributions of this MSc work are:

- C1 – Extensive exploratory evaluation:** we report the results of a detailed exploratory study, using 11 large datasets from physics, finance, transportation, energy, electricity, image, audio and climatic domains, systematically evaluating and validating the ability of the variance preservation and the fractal-based approaches to remove many types of attribute correlations. We show that, at least for large datasets of dimensionality with up to $\sim 1,000$ attributes, our proposed fractal-based algorithm is the best option, being fast, scalable and more accurate to eliminate linear and non-linear correlations. To the best of our knowledge, this is the first work to present a comparative study between variance-preservation techniques and those that are based on the Fractal Theory, by systematically exploring their limitations to remove different correlation types under a variety of circumstances;
- C2 – Novel algorithm:** we propose the new algorithm FReE, a parallel and distributed dimensionality reduction algorithm that uses concepts from the Fractal Theory and Apache Spark to deal with data of high cardinality. FReE implements a novel feature-partitioning strategy that we carefully developed to make it suited for high-dimensionality data processing. To the best of our knowledge, this is the first fractal-based algorithm that is capable of processing billion-scale-elements datasets with hundreds or even thousands of attributes.

1.3 Final Considerations

This chapter presented an overview of our work with a brief description on the facts that motivated it, the problem definition, and our main objectives and contributions. The remaining chapters are organized as follows. Chapter 2 describes the concepts and techniques that are fundamental to our targeted problem. Chapter 3 exposes the main works related to this MSc project. Chapter 4 presents our proposed method FReE, while the methodology used to evaluate the compared dimensionality reduction approaches is given in Chapter 5. Chapter 6 reports the experimental results. Finally, the conclusions and ideas for future works are given in Chapter 7.

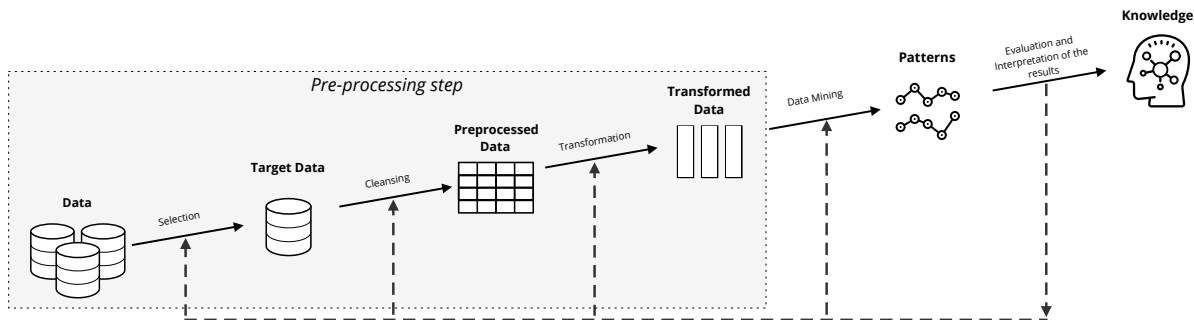
BACKGROUND CONCEPTS

This chapter presents fundamental concepts and techniques related to this MSc work. It begins by describing the process of Knowledge Discovery in Databases (KDD) and the relevance of the preprocessing step in that context. Section 2.2 introduces the dimensionality reduction task with a review of the traditional approaches explored in literature, followed by its use on the context of Big Data. Section 2.3 discusses the distributed processing and some currently related technologies, such as the MapReduce programming model and the Resilient Distributed Datasets (RDD) data structure. The last section presents the final considerations of the chapter.

2.1 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) aims to obtain highly semantic information from raw data (CORDEIRO; FALOUTSOS; TRAINA JR, 2013). Thus, KDD can be defined as “*the process of identifying valid, novel and potentially useful patterns embedded in the data*”(FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). In this process, three significant steps are identified: preprocessing, data mining, and the evaluation and interpretation of results. The preprocessing step can be subdivided into three sub-tasks, as it is illustrated in Figure 1: data selection, cleansing and transformation. During the data selection phase, specific and attractive items from a database are separated for further processing. The cleansing step, also known in some studies simply as preprocessing (ROCHA *et al.*, 2018), aims to correct some inconsistencies found, providing guarantees of reliability in the data that is going to be used for knowledge discovery. After the selection and cleansing steps, the data goes through a transformation step, a process in which the data is usually formatted, reduced or summarized. The data mining stage is responsible for extracting useful and valid patterns, enabling the generation of the knowledge itself. Finally, the last step of the KDD cycle aims to evaluate and perform the interpretation of the patterns obtained through the previous steps, and, if the patterns found have satisfactory results, the knowledge is consolidated. Otherwise, the process returns to previous steps aiming at

Figure 1 – Process of Knowledge Discovery in Databases.



Source: Adapted from [Fayyad, Piatetsky-Shapiro and Smyth \(1996\)](#).

improving the results ([CORDEIRO; FALOUTSOS; TRAINA JR, 2013](#)).

Among the different steps of the KDD process, we highlight the data mining step, which is commonly considered as the core activity of the knowledge discovery process. It involves the application of algorithms that analyze data for the extraction of useful and valid patterns, as mentioned before, following the characteristics of the task to be performed. Those algorithms are classified as: (a) predictive tasks, which aim to find, through the generalization of known examples, a model capable of predicting the value of an attribute, based on the values of other attributes, or; (b) descriptive tasks, which seek patterns that describe intrinsic behaviors of the data ([CLARK; PROVOST, 2019](#); [LESKOVEC; RAJARAMAN; ULLMAN, 2014](#)).

Classification is one of the primary predictive tasks. It considers the existence of a training dataset with objects that were previously classified according to the value of an attribute (target attribute) and a testing dataset to be classified, where the class of each object is still unknown ([CLARK; PROVOST, 2019](#); [CORDEIRO; FALOUTSOS; TRAINA JR, 2013](#)). On those data, algorithms based on decision trees, neural networks, bayesian networks, genetic algorithms, among others, are commonly applied to define rules that represent the relationships between the class attribute and the others. The primary purpose of this task is to predict the target values (the class) of the objects to be classified ([ZAKI; MEIRA; MEIRA, 2014](#)).

Clustering is one of the primary descriptive tasks. It can be defined as “*the process of splitting objects into clusters so that objects in the same cluster show high similarity to each other, and as little as possible similarity to each object in other clusters*” ([HAN; PEI; KAMBER, 2011a](#), p. 443). There is no training dataset; the objects are usually represented in a multidimensional space and a distance function measures the similarity between pairs of objects. Common examples of clustering algorithms are: (a) hierarchical algorithms, which define a hierarchy structure for the data, and it can be started by a single recursive partitioning clustering (top-down), or initially considering that each data object belongs to a distinct cluster, joining each one later (bottom-up), and; (b) partitioning algorithms, which divide n objects into k groups ($k \leq n$), such that each object belongs to a single cluster, and each cluster has at least

one object (LESKOVEC; RAJARAMAN; ULLMAN, 2014; CORDEIRO *et al.*, 2013; HAN; PEI; KAMBER, 2011a; BRYANT; CIOU, 2018).

The KKD process has two significant characteristics: interactiveness and iterativeness. It is interactive because it usually needs user intervention during the sequence of steps to assess the quality and the impact of the patterns encountered in the data mining step, and it is iterative because it is formed by a finite sequence of operations in which each state is dependent on the previous ones (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). In this setting, the preprocessing step has become essential to the KDD process as a whole. At this stage, data is reduced and prepared by cleanse, integration, selection and transformation, directly impacting the quality and performance of later steps, which aim to find patterns. The main problem addressed is the so-called “curse of high dimensionality”, which refers to the fact that increasing the number of data attributes leads to significant performance and accuracy degradation of the existing techniques for manipulation, storage and process of complex data in general, including those that are used in data mining (CHENG; LI; LIU, 2017; GOLAY; KANEVSKI, 2017; ZHANG *et al.*, 2016; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016).

The most successful technique used to minimize the aforementioned problem is dimensionality reduction. It aims to obtain a new set of attributes to represent the data, which should be free of irrelevant, correlated or redundant attributes.

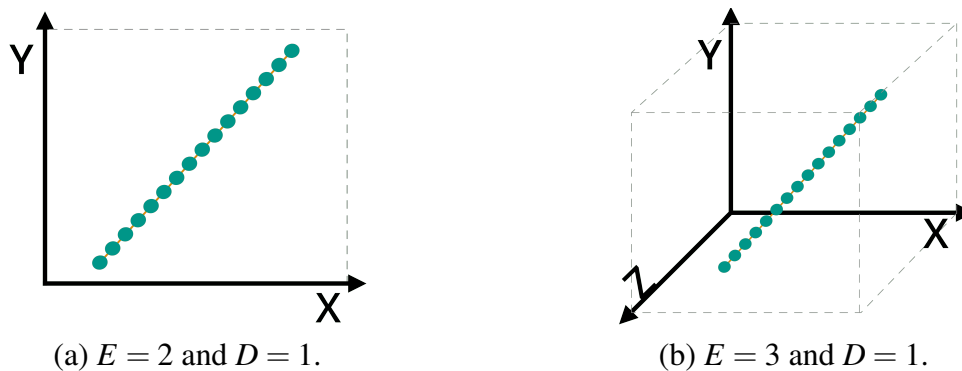
2.2 Dimensionality Reduction in Very Large Datasets

During the preprocessing phase, data is prepared for the pattern extraction step. One of the main problems addressed is the “curse of high dimensionality”, where the number of attributes degrades the performance of algorithms both in runtime and in the quality of results. In data mining and machine learning areas, high dimensionality is a problem faced by both predictive and descriptive tasks (CHENG; LI; LIU, 2017; GOLAY; KANEVSKI, 2017; ZHANG *et al.*, 2016; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016). For example, in predictive tasks, the large amount of attributes increases the search space for the definition of classification models, reducing the accuracy of the discrimination of objects into distinct classes. For descriptive tasks, objects in a high dimensionality space tend to be more sparse, and the distances between any pair of objects tend to be very close, so the objects in the dataset all seem very similar to each other, which makes the aggregation step inefficient and ineffective (CORDEIRO *et al.*, 2013).

The primary technique used to address this problem is dimensionality reduction, which aims to remove redundant information by mapping the original data space into other space of lower dimensionality (GOLAY; KANEVSKI, 2017). In general, the use of a dimensionality reduction method is feasible, because real-world datasets are usually characterized by non-uniform distributions and there exist correlations between the attributes that form the dataset (FALOUTSOS; KAMEL, 1994). In this context, it is essential to note that if two or more

attributes are correlated, there is a mapping capable of determining the value of one of the attributes based on the others, or there is a small number of values that it can assume (SOUSA; AL., 2007). Dimensionality reduction techniques are subdivided into two categories: feature selection and feature extraction. While the former selects the most relevant attributes among the original ones, thus preserving the semantics of the data, the latter creates a reduced set of new attributes to better represent the data by a combination of the original ones, however, losing the original meaning of the attributes (FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016).

Figure 2 – Points over a line with their embedded and intrinsic dimensionalities.



Source: Elaborated by the author.

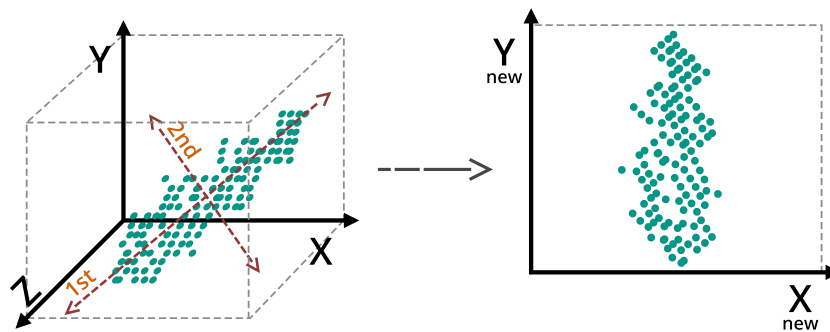
Here, two concepts are fundamentals (TRAINA JR *et al.*, 2010; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016): the **Embedded Dimensionality E**, which is the total number of attributes, and; the **Intrinsic Dimensionality D**, which is the minimum number of attributes required to lossless represent a dataset, regardless of the space in which it is embedded. For example, Figure 2 shows that points over a line have an intrinsic dimensionality $D = 1$, regardless of their embedded dimensionality E . Note that the new attribute Z in Figure 2b does not alter the intrinsic dimensionality, as it does not add extra information by being correlated with X and Y .

Although there exist many works in the literature covering dimensionality reduction, there are still a few algorithms that enable large-scale data analysis in an effective way (FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016). Due to the exponential growth in the amount of data generated in the era of Big Data, the scalability of most algorithms is inadequate (LI *et al.*, 2017). In many scientific and commercial applications, the amount of data is measured in the scale of Terabytes or even Petabytes. In this context, to enable Big Data analysis, massively parallel processing has been used in recent dimensionality reduction algorithms, such as (BALCAN *et al.*, 2016; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016; ELGAMAL *et al.*, 2015). Besides, another fact that makes dimensionality reduction in Big Data attractive is that, in addition to providing improvements in the effectiveness and efficiency of data mining activities, it reduces the amount of data that needs to be manipulated (LI; LIU, 2017). The following subsections present the main approaches to dimensionality reduction that are explored in the literature, and also describe concepts of the Fractal Theory applied to that context.

2.2.1 Dimensionality Reduction based on Data Variance

Variance is a measure that denotes the dispersion of one data distribution of interest. Thus, a small variance indicates that the data tends to be very close to the mean, while a large variance indicates that the data tends to be spread out over a wide range of values (HAN; PEI; KAMBER, 2011b). Many dimensionality reduction techniques use this concept to quantify the capability of attributes to represent information (HAUSER; EFTEKHARI; MATZINGER, 2018; BALCAN *et al.*, 2016). PCA, SVD and KPCA are the most popular ones, being largely used in image processing, data visualization, information retrieval and the like (ELGAMAL *et al.*, 2015; HAUSER; EFTEKHARI; MATZINGER, 2018; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016; BALCAN *et al.*, 2016; DING *et al.*, 2011). In general, these approaches aim at projecting the data into one space of lower dimensionality by finding axes that maximize the data variance. See Figure 3 for an example, where one new axis is initially defined so that it has the most variance of the points, *i.e.*, it is responsible for the maximum variability of the data; then, other axes are adjusted with the restrictions of maximizing the remaining variance and being orthogonal with those axes that were previously defined. Unfortunately, PCA and SVD are limited to the extraction of information based only on linear projections of the original space (ELGAMAL *et al.*, 2015; HAUSER; EFTEKHARI; MATZINGER, 2018; DING *et al.*, 2011). Thus, other variance-preservation methods have been developed to remove non-linear correlations (BALCAN *et al.*, 2016).

Figure 3 – Example of variance-based feature extraction: 3-dimensional points are projected into a 2-dimensional, newly transformed feature space that maximizes variance.



Source: Elaborated by the author.

The state-of-the-art one is KPCA (BALCAN *et al.*, 2016). It aims at removing non-linear correlations through a non-linear transformation of the original space with kernel methods, by extracting the axes that maximize the data variance. It is unfortunate, however, that KPCA requires prior information about the non-linear mapping of the data; it forces the user to choose a kernel method that is appropriate to detect the existing correlations. Since there are several options, *e.g.*, polynomial kernels, Gaussian kernels and cosine kernels, the challenge is to find the appropriate kernel for each dataset. For low dimensional problems, it might not be challenging, but the task becomes harder as the dimensionality increases. This fact limits KPCA's usability

for many real-world applications.

In fact, some datasets require more than one kernel configuration, which can be either with regard to the kernel type or even regarding variations of the kernel's parameters; so, it is then necessary to run KPCA many times with different kernel configurations to detect all correlations. Equation 2.1 describes this process, where $f(\cdot)$ is a dimensionality reduction function, $kernel$ s is an array of kernel functions and κ is the number of kernels. Note that we consider variations of the same kernel's parameters as new functions; for example, for a polynomial kernel we can have a new kernel with degree 2, another one with degree 3, and so on.

$$f(data, \kappa) = \begin{cases} f(KPCA(data, kernels[\kappa]), \kappa - 1) & \text{if } \kappa > 1 \\ KPCA(data, kernels[1]) & \text{if } \kappa = 1 \end{cases} \quad (2.1)$$

Although KPCA has distributed implementations (BALCAN *et al.*, 2016), it still has a high computational cost due to a quadratic space complexity and cubic time complexity regarding the number of objects. When processing datasets of high cardinality, it is therefore common to sample points rather than to build the kernel matrix from the whole dataset. Nevertheless, statistical principles indicate that the sample size must grow together with the size of the full dataset, so the use of KPCA in sets of millions or billions of objects tends to be impractical, even when only one kernel configuration is enough to spot all correlations.

2.2.2 Fractal-based Dimensionality Reduction

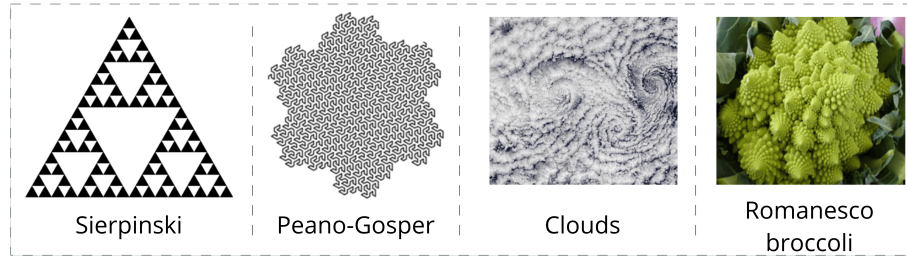
A Fractal is an object that presents exact or statistically approximated similarity when analyzed in different resolutions (SCHROEDER, 1991). Figure 4 shows well-known examples of synthetic and real fractals with exact and statistical self-similarity, such as the Sierpinski triangle and the Peano-Gosper curves. The Sierpinski triangle, for example, is constructed by employing a recursive process that is theoretically infinite; its structure is repeated in different scales.

In the context of data management, Faloutsos and Kamel (FALOUTSOS; KAMEL, 1994) and several subsequent studies (TRAINA JR *et al.*, 2010; FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016; ZHANG *et al.*, 2016) show that real-world datasets commonly behave like fractals, *i.e.*, the spatial object formed by all data points exhibits exact or statistical self-similarity. These works use the concept of **Correlation Fractal Dimensionality D_2** to estimate the intrinsic dimensionality of the data, taking into account the effects of any polynomial or even non-polynomial correlation that may exist among the attributes.

Observation 1. Fractals commonly have unusual and paradoxical properties, which determine that they cannot be considered Euclidean objects with discrete dimensionality. Thus, it is feasible to consider a fractional dimensionality (MANDELBROT; FREEMAN; COMPANY, 1983).

The Box-Counting approach allows computing D_2 with linear complexity on the data size (TRAINA JR *et al.*, 2010; MANDELBROT; FREEMAN; COMPANY, 1983). Following

Figure 4 – Synthetic and real-world fractals with exact or statistical self-similarity.



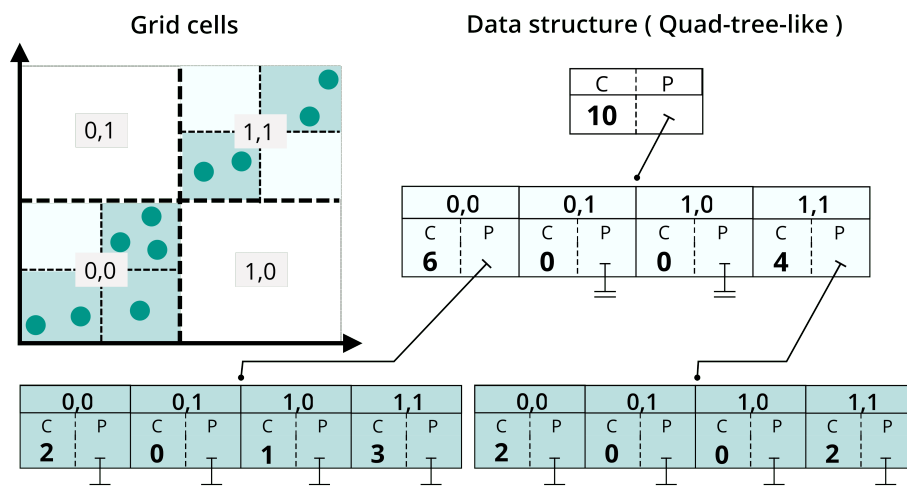
Source: Elaborated by the author.

Definition 1 and Figure 5, it lays hyper-grids with different side sizes over the dataset’s feature space; then, it counts the number of points in each grid. In Equation 2.2, $[r_1, r_2]$ is a range of distances that is representative for the data, r is the side size of the cells in a hyper-grid and $C_{r,i}$ is the count of points in the i^{th} cell of size r . The fractal dimensionality D_2 is the derivative of $\log(\sum_i C_{r,i}^2)$ with respect to $\log(r)$. As we assume self-similar datasets, this derivative results in a constant value. Thus, the dataset’s D_2 is obtained by plotting the sum of squared occupancies in log-log scales for distinct values of r , and capturing the slope of the resulting line.

Definition 1 (Correlation Fractal Dimensionality D_2). Given a dataset that exhibits fractal behaviour, *i.e.*, presents self-similarity in the range of scales $[r_1, r_2]$, its Correlation Fractal Dimensionality D_2 is defined as:

$$D_2 \equiv \frac{\partial \log(\sum_i C_{r,i}^2)}{\partial \log(r)} \quad r \in [r_1, r_2]. \quad (2.2)$$

Figure 5 – Ten points in a two-dimensional dataset divided by hyper-grids, and the corresponding hyper-quad-tree-like structure with two resolution levels.



Source: Adapted from Fraideinberze, Rodrigues and Cordeiro (2016).

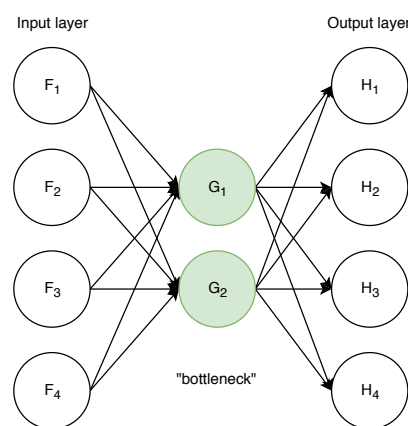
For example, let us consider a dataset with two uncorrelated features: F_1 and F_2 . Its fractal dimensionality is therefore $D_2 \simeq 2$. If we insert one new feature $F_3 = F_1 + F_2$, the value of D_2 will remain close to 2, since F_3 is correlated with the other features.

Concepts from the Fractal Theory have been successfully applied to overcome many problems in data analysis and knowledge discovery. For example, they have been used in join selectivity estimation (FALOUTSOS *et al.*, 2000; BÖHM, 2000; BAIOCO; TRAINA; TRAINA JR, 2007), clustering and classification (BARBARÁ; CHEN, 2000; CORDEIRO *et al.*, 2013), time series forecast (CHAKRABARTI; FALOUTSOS, 2002), data stream forecast and analysis (NUNES *et al.*, 2013; BONES; ROMANI; SOUSA, 2016), dimensionality reduction (FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016; GOLAY; KANEVSKI, 2017; TRAINA JR *et al.*, 2010; ZHANG *et al.*, 2016) and the like.

2.2.3 Other Dimensionality Reduction Approaches

Among the unsupervised dimensionality reduction approaches, there are also those ones based on AutoEncoders. As it is depicted in Figure 6, an AutoEncoder is a trained neural network that attempts to reconstruct the input data after undergoing a compression process. It is divided into two steps: encoder and decoder. The encoder is responsible for compressing the input information in a different latent space by using one specific activation function. The decoder, in turn, does the reverse work and reconstructs the original information, by transforming the latent space created by the encoder into the original information space (FOURNIER; ALOISE, 2019; PETSCHARNIG; LUX; CHATZICHRISTOFIS, 2017).

Figure 6 – Traditional AutoEncoder representation.



Source: Elaborated by the author.

For dimensionality reduction, AutoEncoders are trained with both the encoder and decoder processes, but the output layer is discarded and the encoder output, represented by the “bottleneck” in Figure 6, is treated as the data projection (FOURNIER; ALOISE, 2019; PETSCHARNIG; LUX; CHATZICHRISTOFIS, 2017). For example, in Figure 6 the input layer

receives features F_1 , F_2 , F_3 and F_4 , and after the encoder process, two new features G_1 and G_2 compose the newly transformed space. This approach, although being capable of detecting non-linear correlations among the attributes, has some disadvantages: (a) it requires the user to set a specific number of dimensions for the final output, which is unfeasible in most of the real-world cases; (b) it is specialized only for the dataset that it was trained, *i.e.*, for each dataset it must go through the whole training process again (FOURNIER; ALOISE, 2019), and (c) it is unable to process large datasets, since, to the best of our knowledge, it has no distributed implementation.

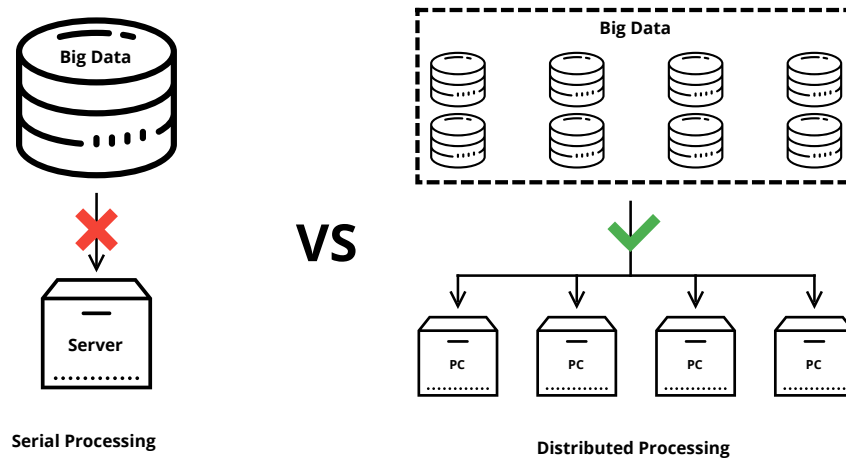
In addition to the approaches detailed in the previous subsections, other dimensionality reduction approaches can be found in the literature, such as those based on the Rough-Set Theory, which generally deal with the manipulation of uncertain information and missing data (CHEN *et al.*, 2016). Also, some strategies rely entirely on machine learning algorithms, such as genetic algorithms, random forests, decision trees and so on (SAYED *et al.*, 2019; SAIDI; NCIR; ESSOUSSI, 2018). However, despite the qualities of those strategies, they are based on supervised learning and consequently require interaction with users, which makes their use unfeasible in most real contexts. For this reason, we consider that these works are outside of the scope of this MSc work, which focuses on unsupervised dimensionality reduction, and therefore, their concepts are not deeply explored here.

2.3 Distributed Processing

The need to process large amounts of data, in the scale of Terabytes or even Petabytes, has encouraged the use of parallel and distributed applications to maximize the efficiency and competitiveness of solutions designed for data analysis (LI *et al.*, 2017). In the dimensionality reduction context, there is a need to develop efficient solutions that handle large amounts of data. As a consequence, distributed processing has become a viable alternative to meet such needs.

The evolution of computer architecture has led to the development of machines capable of processing data very quickly. However, such evolution has a physical limit on the increase of resources that provide, for example, runtime processing. Moore's Law affirms that there is a limit to the development of computer processors (MOORE, 1998). Vertical scaling consists of the improvement of individual machines, aiming at the optimization of resources such as memory, processing power and storage (ERL; PUTTINI; MAHMOOD, 2013). This procedure, as mentioned before, has physical limitations and is commonly expensive, resulting in the need to build systems that work in a distributed manner, *a.k.a.*, horizontal scaling, using the resources of several cheaper and conventional machines (RAO *et al.*, 2019; SINGH; REDDY, 2015).

Distributed computing benefits from the use of several independent devices, which are interconnected through a computer network, allowing the share of resources to use them optimally for the tasks to which they are associated. Therefore, for processing large volumes of data, as it

Figure 7 – Serial Processing *versus* Distributed Processing.

Source: Elaborated by the author.

is illustrated in Figure 7, distributed computing is more efficient, thus reducing/eliminating the aforementioned limitations. Several tools use distributed and parallel processing for extensive data analysis, such as Graphics Processing Unit (GPU) (GUILLÉN *et al.*, 2014) and parallel programming models, like Message Passing Interface (MPI) (ASFOOR *et al.*, 2014). However, most existing tools require the programmer to implement complex parallelism-related procedures. In this setting, the MapReduce programming model has emerged to make the task easier.

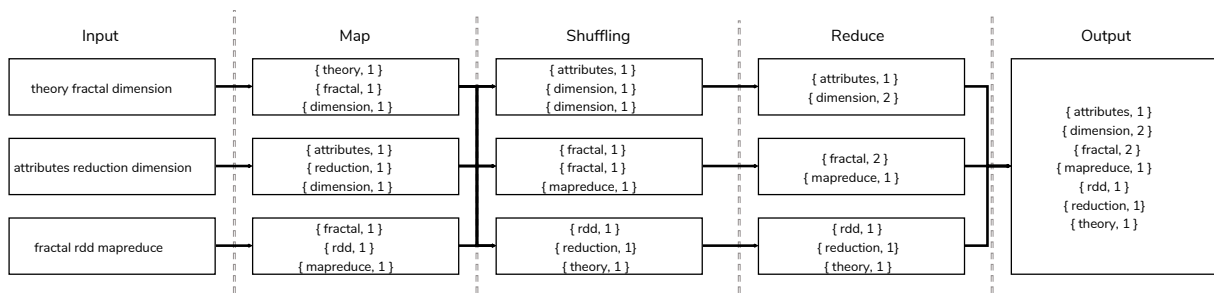
2.3.1 MapReduce Programming Model

MapReduce is a parallel programming model that was initially adopted by Google¹, being introduced to the community in 2004 by researchers Jeffrey Dean and Sanjay Ghemawat (DEAN; GHEMAWAT, 2008) for processing large amounts of data in an “uncomplicated way”. The model provides solutions that hide from the programmer the complexity related to several aspects of parallelism, such as data storage, distribution, replication, fault tolerance, load balancing, among others. It consists of basically three processing stages: (i) Map stage, which is responsible for transforming input data into $\{key, value\}$ pairs; (ii) Shuffling stage, which is responsible for transferring the output generated by the mappers to key-based reducers, and; (iii) Reduce stage, which is responsible for receiving the pairs emitted by mappers, processing the data and generating a final output.

Figure 8 illustrates a typical MapReduce job, where a dataset is stored in a distributed file system, divided into parts, in such a way that there is no overlap of data. These parts are commonly known as chunks of data. The **Map** function takes each chunk of data as input, generates intermediate data with the format $\{key, value\}$, and stores it on the local machine. The process code of the Map must be specified by the programmer, as well as how each data item

¹ www.google.com

Figure 8 – Counting words example in MapReduce model.



Source: Elaborated by the author.

will be represented in the key and value model. Figure 8 demonstrates a classic text-processing example of the model, in which the Map step is responsible for separating each word, producing data where the key is the word and the integer 1 is the value. In the **Reduce** step, the programmer is also responsible for the implementation code that processes the data. The input of this step is the intermediate data sent in the previous step, and through the key, a grouping is performed, transforming the received dataset into a smaller one, depending on the implementation made by the programmer. Following the example illustrated in Figure 8, the key pairs and values generated by the Map process are grouped by summing the values with corresponding keys, which results in the count of the words present in the original dataset.

In 2003, Google presented the Google File System (GFS), a distributed and scalable file system for large amounts of data that provides fault tolerance while running on low-cost hardware. In the year 2006, the Apache community developed the framework Apache Hadoop², an open-source tool whose purpose is to facilitate distributed processing through the MapReduce model (NANDIMATH *et al.*, 2013; WHITE, 2012). Apache Hadoop provides the Hadoop Distributed File System (HDFS) and HBase, which are tools for storage and manipulation of semi-structured data, and PIG, a high-level language for data analysis.

Although it is a widely used tool in the literature with good results, as in (RADENSKI; EHWERHEMUEPHA, 2014; CORDEIRO; FALOUTSOS; TRAINA JR, 2013), the Apache Hadoop implementation of the MapReduce model has a main drawback: slow processing of tasks that need to perform various Map and Reduce steps, due to disk manipulation for large temporary files after each step performed, which is intrinsic to the tool. In this sense, the concept of Resilient Distributed Datasets was introduced to reduce this limitation. It is described in the next subsection.

² <<http://hadoop.apache.org>>

2.3.2 Resilient Distributed Datasets

According to Zaharia *et al.* (2012), a Resilient Distributed Dataset (RDD) is a data structure used in the Apache Spark³ framework, whose central focus is to give preference to store data in RAM and to postpone the execution of operations, *a.k.a.*, lazy evaluation, until the data produced by them is needed. Intuitively, an RDD can be viewed as a database composed of any data, *i.e.*, structured, semi-structured or even fully unstructured data.

The concept of RDD emerged from the Apache Spark framework, which extends the MapReduce programming model, popularized through Apache Hadoop. It minimizes some limitations identified in Apache Hadoop, such as storing large temporary files on disk, thus delivering better performance when compared through some analysis and processing tasks, as in Gu and Li (2013), and more efficiently supporting other kinds of computation, including data stream processing and iterative dataset queries (KARAU *et al.*, 2015).

RDDs are immutable, and while it is supposedly possible to modify an RDD with a transformation, the result of this transformation is a new RDD; the original one remains unchanged. An RDD supports two types of operations:

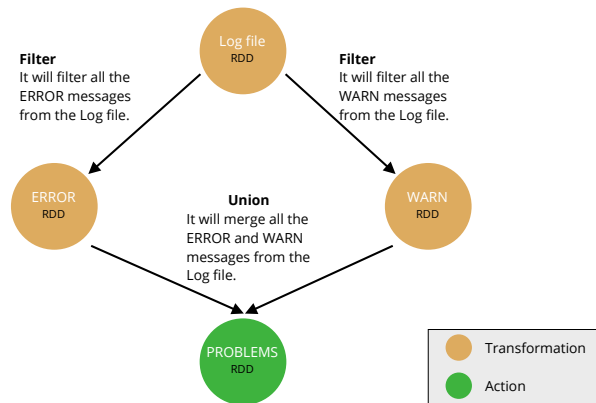
- **Transformation** - It returns not a single value but a new RDD. Nothing is actually executed when the transform function is called; it only indicates how to transform a received RDD as input to generate a new output RDD. Examples of transform functions are `map`, `filter`, `flatMap`, `groupByKey`, `reduceByKey`, `aggregateByKey`, `pipe` and `coalesce`;
- **Action** - An operation to perform immediately on an incoming RDD to get values. When an action function is called on an RDD object, all the transformations required to generate the RDD are effectively executed, and the value is returned. Some of the operations are: `reduce`, `collect`, `count`, `first`, `take`, `countByKey` and `foreach`.

Apache Spark is resilient to failures due to its system called lineage graph, which is exemplified in Figure 9. It stores the dependencies of each action and transformation to compute a particular on-demand RDD and to retrieve lost data if any part of RDD is lost.

As noted earlier, transformations in Spark are lazy-evaluated and therefore are not immediately executed until an action is requested. The framework uses such behavior to reduce the amount of data sent by the grouping operations. Figure 9 illustrates how lazy-evaluation works through an example for ERROR and WARN message filtering in a log dataset. Circles, that represent transformations, take effect only after an action is performed, in this case, the union of two RDDs. By definition, each transformation to an RDD will be performed whenever an action operation is requested on the transformation, but there is a possibility of persisting temporary data in both main memory and disk to speed up data access time if the same RDD needs to be used several times.

³ <<https://spark.apache.org>>

Figure 9 – Lineage graph example: Log files filtering.



Source: Elaborated by the author.

Due to its simplicity, scalability and efficient operations, Apache Spark is an auspicious tool for large-scale data analysis and processing (GU; LI, 2013; ZAHARIA *et al.*, 2012). Also, it has been used to good effect in a variety of applications (RATHORE; AHMAD; PAUL, 2016; ALSHEIKH *et al.*, 2016), many of them perform better, according to the tool’s official website reaching workloads up to 100x faster when compared with similar tasks using Apache Hadoop.

2.4 Final Considerations

This chapter presented several fundamental concepts and techniques related to the problem explored in the MSc work. The KDD process was described, focusing on the importance of the preprocessing step for the results’ quality obtained in the process as a whole. A specific problem that was explored in the preprocessing step is the “curse of high-dimensionality”, which is usually minimized through dimensionality reduction techniques. Unfortunately, the existing techniques are often ineffective as they fail to detect non-linear correlations or even linear correlations in Big Data, mainly due to substantial scalability limitations. For this reason, concepts of Fractal Theory applied to data analysis can reduce both problems presented. Besides, it has been reported that due to the increasing amount of data generated, efficient-distributed-processing demands arise for data analysis through massively parallel processing programming models and resilient and distributed datasets (RDDs) through Apache Spark. In this way, it is possible to give preference to the storage of data in main memory instead of the hard disk, besides delaying the effective execution of operations, aiming at agility in processing and minimizing the communication of data packets in the network.

RELATED WORKS

In the previous chapter, we presented concepts that are essential for this work, such as variance preservation approach to dimensionality reduction, concepts of the Fractal Theory, the parallel programming model MapReduce and Resilient Distributed Datasets from Apache Spark that support the development and deployment of complex algorithms for large clusters of computers. This chapter presents relevant works found in the literature for dimensionality reduction tasks; it is subdivided into two main sections, the first one describes techniques to dimensionality reduction that use the approaches explored in this work and the last one describes works focused in comparative exploration on different approaches.

3.1 Unsupervised Dimensionality Reduction

In this section, recent works designed to analyze large volumes of data are described; many of them are based on algorithms described in the classic approaches already described in the background chapter, such as techniques based on PCA and SVD. Besides, as presented in the background chapter, in the literature, there are other unsupervised dimensionality reduction works, which are based on techniques such as genetic algorithms and neural networks. However, the scope of this MSc project is aimed at comparing techniques based on data variance and techniques based on the Fractal Theory. Therefore, these other dimensionality reduction approaches are not explored in-depth in this work. In this sense, this section is divided into two parts, the first, [subsection 3.1.1](#), describes related works based on data variance, and the second, [subsection 3.1.2](#), describes related works that are based on the Fractal Theory.

3.1.1 *Approaches based on Data Variance*

The PCA and SVD techniques are prevalent in the literature for attribute extraction based on data variance, and they are considered valuable tools in several areas, such as image processing, data visualization, information retrieval and dimensionality reduction ([ELGAMAL](#)

et al., 2015; HAUSER; EFTEKHARI; MATZINGER, 2018). They are statistical procedures that aim to find a new coordinate system, based on the linear transformation of the original coordinate system formed by the dataset to find orthogonal directions of maximum variance. There is a vast amount of work that is based on these two techniques. For example, the SVD technique is the basis for the work of Ding *et al.* (2011), which uses *Message Passing Interface* and the ARPACK library; unfortunately, the technique has cubic complexity concerning the required data dimensionality. For PCA, there is a Spark library MLlib¹ which implements the algorithm on the MapReduce model and takes advantages of RDDs concepts. However, the methods only eliminate linear correlations between attributes, and therefore, they are ineffective in reducing dimensionality in datasets characterized by many non-linear correlations. Also, as they are feature extractors, the algorithms do not preserve the original meaning of the attributes.

Ordozgoiti, Canaval and Mozo (2015) proposed the Parallel Unsupervised Feature Selection (PUFS) algorithm that is based on the Column Subset Selection Problem (CSSP), which selects the attributes incrementally following their relevance by verifying the variance of the data, using the SVD technique, when considered distinct subsets of attributes. Unfortunately, the algorithm can only detect linear correlations between attributes and requires the user to enter/guess the k amount of relevant attributes to keep, which may make the use of this technique unfeasible as such information is generally unknown for most real datasets. Also, the algorithm has cubic complexity concerning the k number of attributes and therefore tends to be unable to process a dataset of high-dimensionality.

Mahout-PCA² is a technique based on a variant of SVD called Stochastic Singular Value Decomposition (SSVD). Mahout-PCA uses a modular framework that finds an approximate version of SVD in a distributed way to compute PCA (HALKO, 2012). The technique was designed in Mahout, which is a collection of machine learning algorithms implemented on top of Apache Hadoop. SSVD uses a random sampling approach to compute an approximated factorization matrix, indicating that there may be variances in the result extracted by the PCA. Also, because it is a technique for extracting attributes, the new extracted set does not retain the original meaning of the attributes that are part of the input dataset.

Scalable Principal Component Analysis (sPCA) (ELGAMAL *et al.*, 2015) is a technique based on the Probabilistic Principal Component Analysis (PPCA) (TIPPING; BISHOP, 1999), designed in the *MapReduce* model and implemented using the Apache Spark tool. sPCA implements optimizations to perform operations on large arrays efficiently, taking advantage of array sparsity, and minimizing the volume of intermediate data created. Unfortunately, both sPCA and PPCA are limited to detecting only linear correlations, apart from the fact that they are extracting methods and eliminate the original meaning of the attributes.

There is also a variation of PCA, called Kernel Principal Component Analysis (KPCA)

¹ <<https://spark.apache.org/docs/2.3.1/ml-guide.html>>

² <<http://mahout.apache.org>>

(SCHÖLKOPF; SMOLA; MÜLLER, 1998), which allows detecting nonlinear correlations between attributes through kernel functions. Balcan *et al.* (2016) proposed a version of KPCA capable of analyzing multidimensional data in a distributed model running on multiple machines through sampling techniques. The method includes different heuristics that aim to reduce the cost of communication between cluster machines, such as running PCA locally on each slave machine before sending the results to the master machine. Fraideinberze, Rodrigues and Cordeiro (2016) performed a series of experiments using the aforementioned distributed version of KPCA, which was unable to analyze databases with millions of elements in a viable time, exceeding a threshold of 3 days of analysis in a cluster composed of 20 nodes, each containing 8 cores, 14GB of RAM and 600GB of the disk, without response. In this sense, the algorithm proved to be unfeasible to process data with high cardinality, for example, billions of elements.

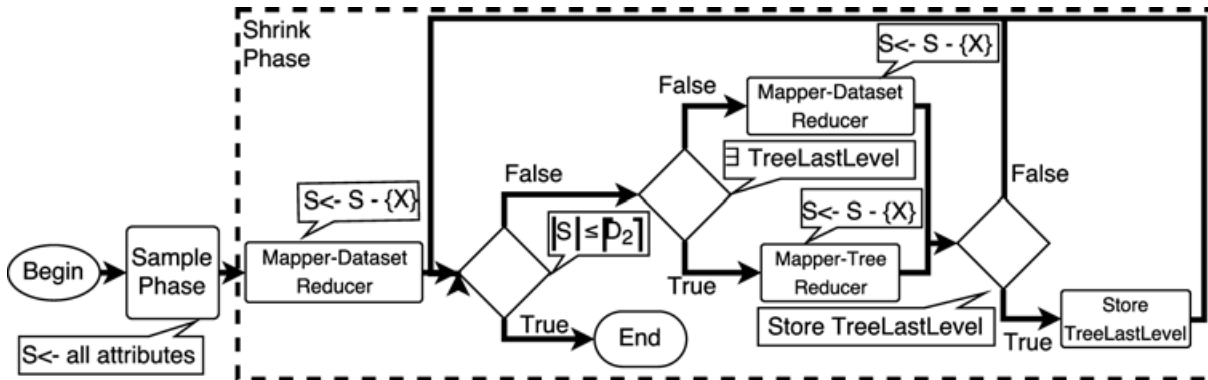
In spite of the many qualities present in the aforementioned algorithms, most of them have the following limitations: (i) they find only linear correlations between the attributes, or; (ii) they have superlinear scalability regarding the data dimensionality or cardinality. Additionally, except for PUFS, all the other algorithms are feature extraction methods; they create new attributes based on the combination of the original attributes and thus return attributes that do not have the original meaning of the analyzed dataset.

3.1.2 Approaches based on the Fractal Theory

In (TRAINA JR *et al.*, 2000; TRAINA JR *et al.*, 2010), the algorithm for feature selection *Fractal Dimensionality Reduction* - FDR is proposed. Using a differentiated approach, it takes advantage of concepts from the Fractal Theory. The technique uses the Fractal Dimensionality D_2 to infer the intrinsic dimensionality of the dataset and remove the least relevant attributes, which, when eliminated, cause less impact on the fractal dimensionality. Because it is based on the Fractal Theory, FDR can detect and eliminate attributes with linear and non-linear correlations in an unsupervised way, that is, in multidimensional datasets with or without class information. To the best of our knowledge, FDR was the first feature selection algorithm based on fractals presented in the literature; after its proposal, other variations of algorithms appeared, such as (ZHANG *et al.*, 2016; GOLAY; KANEVSKI, 2017). Although those techniques provide excellent results, being able to detect linear and non-linear correlations with linear scalability regarding the number of points in the dataset, they were designed to work on a single computer core, and are therefore unable to process large volumes of data.

Fraideinberze, Rodrigues and Cordeiro (2016) proposed the CurlRemover algorithm for feature selection using concepts from the Fractal Theory. It works through a parallel approach under the MapReduce programming model in Apache Hadoop. It is important to emphasize that, as it happens with the FDR algorithm, CurlRemover is also seen as a relevant precursor to the present MSc work, since it uses several techniques that were explored and improved in this new proposal. Therefore, it is described with greater emphasis in the following.

Figure 10 – General pipeline of CurlRemover.



Source: Fraideinberze, Rodrigues and Cordeiro (2016).

As it is illustrated in Figure 10, the algorithm has two main steps: *Sample* and *Shrink*. The first step, *Sample*, is responsible for analyzing a small data sample extracted from the input dataset aimed at obtaining a rough estimate of the relevance of its attributes. The second step, *Shrink*, computes the final set of relevant attributes from the complete analysis of the input dataset, using the attributes obtained in the first step to minimize the computation, disk access and data traffic on the network between cluster machines, in addition to providing better workload balancing between these machines.

The second step of the algorithm is supported by concepts from the Fractal Theory to search for one irrelevant attribute at a time, in ascending order of relevance, until the $E - [D_2]$ least relevant attributes are identified. As it is described in section 2.2, E represents the embedded dimensionality of the dataset while D_2 is the corresponding Correlation Fractal Dimensionality. As it happens in algorithm FDR, a multidimensional quad-tree data structure – see illustration in Figure 5 – is used to implement the BoxCounting approach, enabling the computation of D_2 with linear scalability regarding the cardinality of the dataset. Specifically, the feature space is recursively divided into cells of different side sizes, each storing one ID , the count of points that are in the space of that cell, and a pointer to the next level of the tree (FRAIDEINBERZE; RODRIGUES; CORDEIRO, 2016).

The tree-based data structure is built in the main memory. Each node is implemented through a linked-list or a structure of type $\{key-value\}$, like a red-black tree, using the ID as the key and counts of points as the values. Each level of the tree has at least one point per cell and represents the value of r , which is the size of the box defined in Equation 2.2 of subsection 2.2.2. It is worth noting that, although the memory usage is linear on the data cardinality, the amount of memory available for each mapper is usually very limited, and generally, it is not enough to build the entire tree. Therefore, the authors address this problem by creating partial trees in the map step and finish building the tree in the reduce step.

Algorithm CurlRemover achieved excellent results in terms of accuracy in the experi-

ments reported in its original publication, being superior to state-of-the-art techniques, such as sPCA and the distributed version of KPCA, described in subsection 3.1.1. In spite of this fact, unfortunately, the technique is not capable of working with large volumes of data, for example, data on the scale of Terabytes. As is was described previously, CurlRemover eliminates one irrelevant dimension at a time, writing and retrieving large temporary files to/from the hard drives. Despite having an internal heuristic that speeds up the process of removing each irrelevant attribute, in the worst case scenario, the dataset is read numerous times during the process, once for each attribute removed. Additionally, CurlRemover was designed to process datasets of medium dimensionality, being unable to process datasets of high dimensionality in a viable runtime and with accurate results.

After systematically reviewing the literature, we conclude that it is possible to find several studies for unsupervised dimensionality reduction based on data variance or on concepts of the Fractal Theory. However, the existing approaches have relevant limitations, whether regarding the inability to analyze large volumes of data, *i.e.*, Terabytes or Petabytes, or with regard to the inability to detect non-linear attribute correlations, among other minor limitations. Figure 11 summarizes the main characteristics of the algorithms described in this chapter, where the red color symbolizes non-satisfiability of the given algorithm concerning the condition imposed in the column and the green color symbolizes the opposite of the red color, that is, the proper functioning of the algorithm under the imposed condition.

Figure 11 – Summary of algorithms.

Technique	Process Terabytes of data	Detect linear and non-linear correlations	Process high dimensionality	Maintains the original meaning of the features
PCA-MLlib				
SVD-MPI				
PCA-Mahout				
sPCA				
PUFS				
KPCA				
FDR				
CurlRemover				

Source: Elaborated by the author.

The characteristics of each technique are summarized in the form of answers to four questions that are essential to data analysis in the context of *Big Data*. As it can be seen, despite the various qualities present in the existing works, to the best of our knowledge, there is no algorithm capable of analyzing *Terabytes* of high-dimensionality data in an efficient and effective way, thus detecting and removing linear and non-linear correlations.

3.2 Comparative Studies

Popular algorithms that attempt to maximize the data variance are the basis for most state-of-the-art methods in unsupervised dimensionality reduction. On the other hand, there are other promising but less explored techniques, like those based on fractals. From this perspective, the need for comparative studies by evaluating their effectiveness and efficiency arises.

In the current literature, there exist such kind of work. Examples are: (GISBRECHT; HAMMER, 2015; YEH *et al.*, 2005; MEIER; KRAMER, 2017; DU, 2018; GOLAY; KANEVSKI, 2017). Despite their many qualities, to the best of our knowledge, no one presents a comparative study between variance-preservation techniques and those that are based on the Fractal Theory, by analyzing their abilities to detect several types of correlations from large amounts of data. For example, Du (2018) presents a comparative study of dimensionality reduction techniques over morphometric data by comparing PCA with four other non-linear approaches. The results show that the non-linear approaches are superior to PCA regarding the preservation of essential information of the dataset analyzed. However, this work does not evaluate any approach based on the Fractal Theory. Also, it only considers a very particular domain and does not test methods well-suited to process very large datasets.

In Golay and Kanevski (2017), a new unsupervised feature selection technique is proposed, and it is compared with state-of-the-art feature selection techniques, including FDR. The algorithms were evaluated with real and simulated datasets, considering their abilities to detect non-linear correlations and the quality of the selected features using the Random Forests classification algorithm (BREIMAN, 2001). Unfortunately, the aforementioned work made a comparison on feature selection techniques according to the results obtained with a machine learning algorithm, which can add bias to the process. Besides, the authors used only small datasets and did not test any feature extraction algorithm.

In this MSc work, we contribute to the state-of-the-art by comparing the variance-preservation approach for unsupervised dimensionality reduction with the fractal-based one, mainly focused on large datasets of high-dimensionality, *i.e.*, data with millions of objects and up to ~ 1000 axes. We systematically explore their limitations to remove different types of correlations, under a variety of circumstances.

3.3 Final Considerations

This chapter described state-of-the-art techniques used for the task of dimensionality reduction in an unsupervised way, divided into two categories: (i) based on data variance, and; (ii) based on the Fractal Theory. Additionally, some works that compare dimensionality reduction techniques have been described, concluding that, to the best of our knowledge, this MSc work is the first one that performs an in-depth comparison between the two aforementioned approaches.

The next chapter describes the method that we propose, based on Fractal Theory concepts, which meets all the conditions listed in [Figure 11](#).

PROPOSED METHOD

4.1 Proposed Method

Fractal-based techniques assume that the intrinsic dimensionality of the object described by the dataset is smaller, generally much smaller than that of the space where the object is represented, i.e., the embedded dimensionality. Since this assumption has been valid in the vast majority of real situations, these techniques are useful to indicate a target dimensionality to which the dataset can be reduced, using Fractal Theory techniques or not, thereby diminishing the dimensionality curse. However, as it was described in the previous chapters, the existing approaches for dimensionality reduction have two main limitations:

- **Dimensionality limitation:** they can only process data of medium dimensionality with at most a few tens of attributes;
- **Cardinality limitation:** they are unable to process very large datasets with millions or even billions of objects.

This chapter presents the new algorithm **Fractal Redundancy Elimination (FReE)**. It tackles the dimensionality limitation with a novel feature set partitioning approach that iteratively eliminates redundant features by processing them in separate, small blocks of features, thus making it feasible to work with hundreds or even thousands of them. FReE also takes advantage of the MapReduce model through Apache Spark¹ to efficiently process hundreds of millions or even billions of objects in a resilient and distributed way, so tackling the cardinality limitation.

4.1.1 Dimensionality Limitation

Let us first focus on how to shrink the dimensionality limitation. It happens because the estimation of intrinsic dimensionality is based on numerical processes that measure the slope of

¹ <spark.apache.org>

the cumulative distribution curve of the number of object pairs that are at progressively longer distances from each other, as it was described in Section 2.2.2 from Chapter 2. For real datasets, this curve may be suitably approximated by a line, and its slope is assumed to be the intrinsic dimensionality of the data. When the value of this measurement is small, say less than 6, the error introduced by the numerical slope measurement process is usually negligible. Note that a slope equal to 6 corresponds to more than 80 degrees. However, as the intrinsic dimensionality increases, thus bringing the inclination angle to near 90 degrees, small errors in fitting the line to the distance distribution curve lead to significant errors in the measured value. This characteristic makes the process unstable and reduces the reliability of the result.

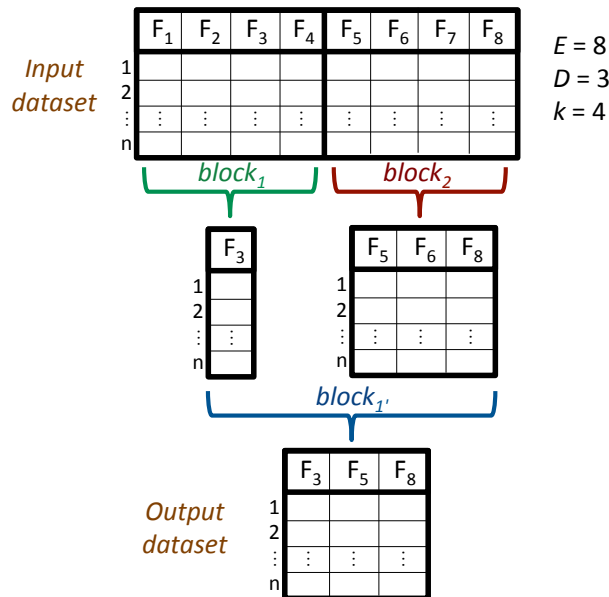
The existing techniques significantly reduce the original data dimensionality. However, they are still limited to the fact that each new attribute added, even if it is highly correlated to the others, causes a small increase in the intrinsic dimensionality. When the embedded dimensionality of a dataset is on the order of dozens of attributes, these techniques allow reducing the dimensionality to values typically smaller than 10. However, when the number of attributes rises to the hundreds or thousands, the contributions of the attributes to the intrinsic dimensionality, even if individually small, accumulate to raise it to the tens, thus causing the aforementioned instability in the numerical slope measurement process. As a consequence, techniques like FDR and its sequels are only suited to process data with intrinsic dimensionality up to nearly 10 attributes, which usually corresponds to embedded dimensionalities on the order of dozens of attributes.

With that in mind, we propose to shrink the dimensionality limitation by using a novel feature set partitioning approach. It is a simple, yet powerful strategy that allows the analysis of hundreds or even thousands of features by iteratively processing k -dimensional data projections, where k represents the maximal embedded dimensionality that can be handled by the existing techniques; so, k is on the order of dozens. Instead of processing all attributes together as the previous works do, we only process at most k attributes at a time, thus iteratively removing redundant attributes until obtaining a single set with less than k attributes to be processed together, so the numerical slope measurement process is always stable and the reliability of the results increases. The user must previously define this parameter k ; we used $k = 30$ in all experiments that we performed, obtaining very good results. Thus, we believe that tuning this parameter according to the dataset is not a problem. In this way, we shrink the dimensionality limitation by being suited to process data with intrinsic dimensionality up to nearly k attributes, *i.e.*, dozens of attributes, which usually corresponds to embedded dimensionalities on the order of hundreds or thousands. Experimental results from 11 real-world datasets with up to $\sim 1,000$ attributes corroborate this fact; see Section 6.1 from the upcoming Chapter 6 for details.

Figure 12 illustrates our proposed feature set partitioning for one example 8-dimensional dataset with intrinsic dimensionality $D = 3$ and cardinality n . Let us consider $k = 4$. Note that we use very low dimensional data in this example to avoid cluttering the illustration; in practice,

data of much larger dimensionality are processed with k on the order of dozens. At first, our algorithm FReE processes in separate two 4-dimensional data projections. Let us name them as $block_1$ and $block_2$. Each $block$ contains k sequential features considering the same order of storage observed in the original data file; for example, $block_1$ is initially composed of features F_1 to F_4 and $block_2$ is composed of features F_5 to F_8 . This process allows FReE to find out that $block_1$ contains one single relevant feature, F_3 . It means that F_3 is correlated with features F_1 , F_2 and F_4 , so the latter ones are discarded. The same process is used to reveal that $block_2$ has three relevant features: F_5 , F_6 and F_8 . Once the two blocks are processed, a new partitioning round is performed on the remaining features following an iterative process that always analyzes blocks with at most k sequential features each, until one single block remains. In our example, the second partitioning round is the last one, since only four features are remaining from the previous round, i.e., $block_{1'}$ with features F_3 , F_5 , F_6 and F_8 . Then, $block_{1'}$ is processed to reveal that feature F_6 is correlated with F_3 , so it is discarded from the final result.

Figure 12 – Example feature set partitioning with embedded dimensionality $E = 8$, intrinsic dimensionality $D = 3$, cardinality n and block size $k = 4$.



Source: Elaborated by the author.

4.1.2 Cardinality Limitation

Now, let us focus on how to shrink the cardinality limitation. It happens because the existing fractal-based approaches for dimensionality reduction were either designed for single-core processing or they perform too many disk accesses to process the data in parallel using a cluster of computers. As a consequence, techniques like FDR and its sequels are unable to process very large datasets with millions or even billions of objects in feasible runtime. With that in mind, we propose to shrink the cardinality limitation by taking advantage of the MapReduce

model through Apache Spark², and carefully designing one solution that prioritizes the use of the cluster's main memory over the secondary memory. It allows our proposed algorithm FReE to process Big Data in a resilient and distributed way efficiently.

Algorithm 1 – Fractal Redundancy Elimination (FReE)

Require: Dataset A , blockSize k , threshold t

Ensure: Dataset A without redundant attributes

```

1: Read dataset  $A$  from the distributed file system and normalize it;
2:  $existsRedundancy = True$ ;
3: while  $existsRedundancy$  do
4:   Split  $A$  into blocks of at most  $k$  sequential features each;
5:   for each block  $block_i$  do
6:      $block_i = remove\_redundancy(block_i, t)$ ; // from Algorithm 2
7:   end for
8:    $A =$  all blocks merged;
9:   if there was no change in  $A$  then
10:     $existsRedundancy = False$ ;
11:   end if
12: end while
13: return  $A$ ;

```

Algorithm 2 – $remove_redundancy()$

Require: Block $block$, threshold t

Ensure: Block $block$ without redundant attributes

```

1:  $E =$  number of attributes in  $block$ ;
2:  $D_2 =$  fractal dimensionality of  $block$ ;
3:  $currentD_2 = D_2$ ;
4:  $existsRedundancy = True$ ;
5: while  $existsRedundancy$  do
6:   for every attribute  $F_i$  in  $block$  do
7:      $partialD_{2,\{block-F_i\}} =$  fractal dimensionality of  $block$  without attribute  $F_i$ ;
8:   end for
9:   Let  $F_a$  be the attribute that leads to the smallest difference  $|currentD_2 - partialD_{2,\{block-F_a\}}|$ ;
10:  if  $partialD_{2,\{block-F_a\}} \leq E \wedge \frac{|D_2 - partialD_{2,\{block-F_a\}}| \times 100}{D_2} < t$  then
11:    Remove attribute  $F_a$  from  $block$ ;
12:     $currentD_2 = partialD_{2,\{block-F_a\}}$ ;
13:  else
14:     $existsRedundancy = False$ ;
15:  end if
16: end while
17: return  $block$ ;

```

Algorithm 1 provides the pseudo-code of FReE. It receives as input one dataset A , the block size k and one additional parameter t . Intuitively, t defines the maximum loss of information

² <spark.apache.org>

that is acceptable after reducing the dimensionality of dataset A . Note that we used fixed values $k = 30$ and $t = 0.5\%$ in every single experiment reported in this monograph, so we believe that tuning these parameters should not be a problem for the user. At first, FReE splits the feature set by assigning at most k sequential features to independent blocks; see Line 4. The blocks are then processed separately in Lines 5 to 8 using the fractal dimensionality D_2 as a tool to select the relevant features within each block, and all relevant features are merged. The algorithm is iterative; that is, it repeats the splitting and processing steps using as input the resulting set of features from the previous iteration until there are no more features to be removed without changing the fractal dimensionality more than threshold t .

Algorithm 2 details our proposed strategy to process each block. At first, FReE computes the fractal dimensionality D_2 of the block; see Line 2. Then, Lines 6 to 8 calculate partial fractal dimensionalities $partialD_{2,\{block-F_i\}}$ by ignoring one attribute F_i at a time from the block. The attribute F_a leading to the smallest difference between the block's current fractal dimensionality and $partialD_{2,\{block-F_a\}}$ is then removed in Line 11 because this attribute is the one that adds the least amount of information to the data. This process is repeated until reaching the loss threshold t , and then the set of relevant features is returned in Line 17.

FReE uses the BoxCounting strategy described in Section 2.2.2 from Chapter 2 to compute the fractal dimensionality D_2 and the partial fractal dimensionalities $partialD_{2,\{block-F_i\}}$. A hyper-quad-tree-like data structure is employed to speed up the process; see Figure 5. We map the dataset points in a $\{key, value\}$ format, where the key is the cell ID , and the value is the number of points in that cell. In this step, for each worker of the cluster, the Map and Reduce processes are performed together through an Apache Spark's hashmap data structure, thus ensuring that there is at most one $\{key, value\}$ pair for each cell in the same worker. Also, the data storage process prioritizes the use of the cluster's main memory over the secondary memory. Those mapped data then go through a global Reduce process, where summations are performed to obtain a single pair for each tree level, with the logarithm of the side size of the cells and the logarithm of the sum of the squared number of points in each cell corresponding to the level.

4.1.3 Computational Complexity Analysis

As is was mentioned before, one of the most significant challenges in the era of Big Data for dimensionality reduction algorithms is having the ability to handle large datasets with appropriate scalability concerning their cardinality. Our algorithm has linear scalability on the number of data objects, being able to process millions or even billions of objects effectively. Here, we analyze the computational complexity of its most computationally intensive steps.

FReE uses the fractal dimensionality D_2 as intrinsic dimensionality information to support the feature selection step, eliminating those attributes that least impact the fractal behavior of the analyzed dataset. This characteristic generates the need to perform the fractal

dimensionality calculation several times for the analyzed dataset, making this a computationally intensive step for the algorithm as a whole. FReE implements the Box-Counting approach to support the fractal dimensionality calculation, which uses a hyper-quad-tree-based data structure as represented in [Figure 5](#) and [Equation 2.2](#) from [Chapter 2](#). In our work, the Box-Counting method took advantage of a HashMap structure with lazy-evaluation operations, which is provided natively by the Apache Spark framework. In its worst-case scenario, the computational complexity is $O(N)$, where N is the number of objects in the dataset. For memory utilization, the maximum amount required to store this structure is reached when only one point occupies each cell on each tree level; therefore, it is also linear to the number of objects on a dataset. For a more detailed analysis, see the works of [TRAINA JR et al. \(2010\)](#), [TRAINA JR et al. \(2000\)](#), and [Fraideinberze, Rodrigues and Cordeiro \(2016\)](#).

Algorithm FReE uses a backward elimination strategy to perform feature selection, as is explained in [Algorithms 1 and 2](#). A partitioning methodology composes this method, which splits the attribute space into blocks of maximum size k , where k is a predefined value and represents the maximum embedded dimensionality that can be handled by the existing fractal-based techniques; this value is on the order of the dozens. As it was explained in the previous section, while redundancy exists in each block of size k , one irrelevant attribute is removed at a time, which is the attribute that least interferes with the fractal dimensionality of the dataset. After removing all redundancies from each block, the blocks are merged and then subdivided into blocks with up to k features again, until there is no more redundancy in the entire dataset.

This strategy benefits not only the quality of the results obtained in high dimensional datasets, since the numerical slope measurement process is always stable, but it also reduces the computational complexity found in state-of-the-art algorithms that are based on the Fractal Theory, such as [TRAINA JR et al. \(2010\)](#) and [Fraideinberze, Rodrigues and Cordeiro \(2016\)](#). For example, considering the worst case scenario of the aforementioned state-of-the-art algorithms, for a dataset with an intrinsic dimensionality equals to 1, the algorithms process it with runtime $O(N \times (\frac{d \times (d-1)}{2}))$, where N is the number of points in the set and d is the number of features. This complexity is the result of the process in which, for each feature to be removed, the fractal dimensionality of the entire set of features is calculated to delete the one that has the least impact on the fractal dimensionality. To calculate the fractal dimensionality D_2 we have a complexity of $O(N)$ and to test all the feature relevance we have the complexity $O(\frac{d \times (d-1)}{2})$.

When using our proposed partitioning strategy, the set of features to be processed is divided into smaller parts with up to k features each, which can be processed in parallel, with l merging operations until there is no redundancy in the set, thus transforming the runtime complexity into $O(N \times ((l \times \frac{k \times (k-1)}{2})))$. Here, l is at most the value of the intrinsic dimensionality of the analyzed dataset and k can be adjusted to a value close to that number.

To summarize the time complexity analysis of algorithm FReE, it is linear on the number of objects N and $O((l \times \frac{k \times (k-1)}{2}))$ on the number of features, where k and l tends to be close to

the intrinsic dimensionality.

4.1.4 Final Considerations

This chapter introduced Fractal Redundancy Elimination - FReE, a new algorithm for selecting features in very large datasets in an unsupervised way. The main innovations in the technique are the proposal of a new feature partitioning approach that is well-suited to deal with high-dimensional data sets, and the use of RDD concepts to take advantage of the speed of data processing in main memory. In the next chapters, we show through an evaluation methodology and results of comparative experiments, that the proposed technique is suitable for many real datasets, obtaining better results than state-of-the-art techniques.

PROPOSED EVALUATION

5.1 Proposed Evaluation

This chapter describes the methodology that we propose to evaluate dimensionality reduction techniques. It includes the materials used and details about the systematic evaluation performed to answer the following questions:

- Q1** In practice, what types of attribute correlations is our fractal-based algorithm capable of removing? What about PCA, SVD and KPCA?
- Q2** What is the influence of increasing the number of redundant attributes regarding the techniques studied?
- Q3** Are the techniques capable of removing redundant attributes that are correlated with more than one other attribute?
- Q4** What is the effectiveness of the techniques when applied to data with different types of correlations, altogether?

For an in-depth evaluation, we studied 11 real datasets from distinct domains. They are summarized in Table 1, and detailed descriptions are found at their original sources; see footnotes at the table. For **reproducibility**, all codes, detailed results and datasets used in this work are freely available for download at: <http://bit.ly/2k5QiQH>.

5.1.1 Evaluation

To perform the evaluation, we initially determined a set of correlation types to be studied, among them: linear, polynomial and non-polynomial correlations. Table 2 lists the correlations used to generate new redundant attributes, including their corresponding equations, where F_β

Table 1 – Summary of datasets used to evaluate the techniques studied.

Dataset	# Points	# Attributes	Domain
Airline ^a	123.5M	21	Transportation
Forex ^b	60.6M	17	Finance
Higgs ^c	11.0M	28	Physics
Hepmass ^c	10.5M	28	Physics
WeatherBR ^b	9.4M	24	Climate
Susy ^c	5.0M	18	Physics
NaturalGas ^d	2.1M	4	Energy
Household ^c	2.1M	13	Electricity
Fonts ^c	0.7M	409	Image
YearPredMSD ^c	0.5M	90	Audio
FMA ^c	0.1M	518	Audio

^a <stat-computing.org/dataexpo/2009/the-data.html>

^b <www.kaggle.com/datasets>

^c <archive.ics.uci.edu>

^d <dataverse.harvard.edu/dataverse/harvard>

is the new attribute to be generated, C_1 and C_2 are randomly predefined constants and F_α is an existing attribute to be randomly chosen from the original dataset.

Table 2 – Equations used to generate redundant attributes.

Correlations	Equations
Linear	$F_\beta \approx F_\alpha \times C_1 + C_2$
Logarithmic	$F_\beta \approx \log F_\alpha \times C_1$
Quadratic	$F_\beta \approx F_\alpha^2 \times C_1 + C_2$
Cubic	$F_\beta \approx F_\alpha^3 \times C_1 + C_2$
Exponential	$F_\beta \approx C_1^{F_\alpha}$
Tenth power	$F_\beta \approx F_\alpha^{10} \times C_1 + C_2$
Hundredth power	$F_\beta \approx F_\alpha^{100} \times C_1 + C_2$
Sinusoidal	$F_\beta \approx \sin F_\alpha \times C_1$

Increasing the number of redundant attributes: We evaluated the techniques by adding m new attributes in each dataset studied, where $m \in \{1, 5, 10, 20, 50, 100, 500\}$. Each new attribute is correlated with a single original attribute employing a specific type of correlation, among those in Table 2. Considering that a good technique should also be robust to the presence of noise, since we assume that real-world datasets commonly have this characteristic – for example, maximum and minimum weather temperatures in a period of time are nearly linearly

correlated, but not exactly correlated – we also evaluated the behavior of the techniques by adding noise in the correlations, and by increasing the amount of noise present in the new redundant attributes. In order to simulate non-exact correlations, each new redundant attribute was “corrupted” with Gaussian noise, where the mean of the noise was fixed at 0, and the standard deviation was progressively set to 0.2%, 1%, 2%, 5% and 10% of the standard deviation of the original feature. For example, suppose we have a dataset with an attribute F_1 . One attribute F_2 linearly correlated with F_1 would be formed from the following equation: $F_2 = (F_1 + \text{noise}()) \times C_1 + C_2$, where $\text{noise}()$ can be a negative or positive value within the established percentage of the standard deviation of F_1 .

Increasing the number of attributes per correlation: We also evaluated the techniques regarding the number of attributes used to generate each new redundant attribute. For example, given a dataset with attributes F_1 and F_2 , a new redundant attribute F_3 could be formed from the two original ones, *e.g.*, $F_3 \approx F_1^2 + F_2^2$. Here, we considered all types of correlations from Table 2, and for each of them, 5 new redundant attributes were added to each dataset that we studied. Each redundant attribute is correlated with n distinct original attributes, where n ranges from 2 up to the original embedded dimensionality E of the dataset, or up to 50 when $E > 50$. We also included noise in the new redundant attributes, where the standard deviation of the noise was progressively set to 1%, 2%, 5%, 7% and 10% of the standard deviation of each original attribute used in each correlation.

Mixed correlations: So far, we focused on evaluating the techniques using a single type of correlation at a time. Nevertheless, real data are likely to present several types of correlations, all together. With that in mind, we repeated the procedures described in the last two paragraphs, but in this time, randomizing the correlation used to generate each new attribute, still keeping the variations in the amount of noise to simulate inexact correlations.

5.1.2 Algorithm settings

The variance-preservation techniques studied are PCA, SVD, PUFs and KPCA with a polynomial kernel function of degree 2, which is one of the most used kernels. Note that PCA and SVD work differently, but they return the same eigenvectors and eigenvalues. Therefore, we refer to them as PCA/SVD in the quality results reported. For these techniques, we used the MLlib implementation¹. Due to the high computational cost of KPCA, we used a kernel matrix approximation obtained from an adaptive sampling technique based on centroid points extracted with the k -means clustering algorithm. The value of k was set to $E \times \log E$, where E is the embedded dimensionality of the dataset; this value was defined according to a suggestion from Balcan *et al.* (2016). As the algorithms are sensitive to data variance, it was also necessary to normalize the datasets. On the other hand, the fractal-based approach used in the evaluation is our algorithm FReE with fixed parameter values, $k = 30$ and $t = 0.5\%$.

¹ <<https://spark.apache.org/docs/2.3.1/mllib-dimensionality-reduction.html>>

5.1.3 Final Considerations

This chapter described our proposed evaluation for unsupervised dimensionality reduction algorithms. The proposal consists of the evaluation of such algorithms through the insertion of redundant attributes in real-world datasets generated from linear, polynomial and non-polynomial correlations. For this study, we used 11 real datasets from different domains. It is noteworthy that the proposed methodology can be used to evaluate any other unsupervised dimensionality reduction algorithm, without the risk of adding bias to the final result, unlike other evaluation methodologies that use the results of machine learning algorithms to evaluate the effectiveness of the dimensionality reduction task. The next chapter presents the results obtained by the approaches studied when submitted to the methodology described in this chapter.

EXPERIMENTAL RESULTS

6.1 Experimental results

In this chapter, we discuss the experiments performed to answer the questions posed at the beginning of Chapter 5. The experiments used a Microsoft Azure cluster with 8 machines: two *masters*, each one with 4 cores, 14GB of RAM and 200GB of disk, and; 6 *workers*, each one with 8 cores, 28GB of RAM and 600GB of disk. We configured the machines with GNU/Linux Ubuntu 16.04 server x64. We also used 6 machines from Amazon Web Services (r5a.4xlarge), each one with 16 cores, 128GB of RAM and 100GB of disk. We thank both Microsoft and Amazon for providing us with free access to these resources.

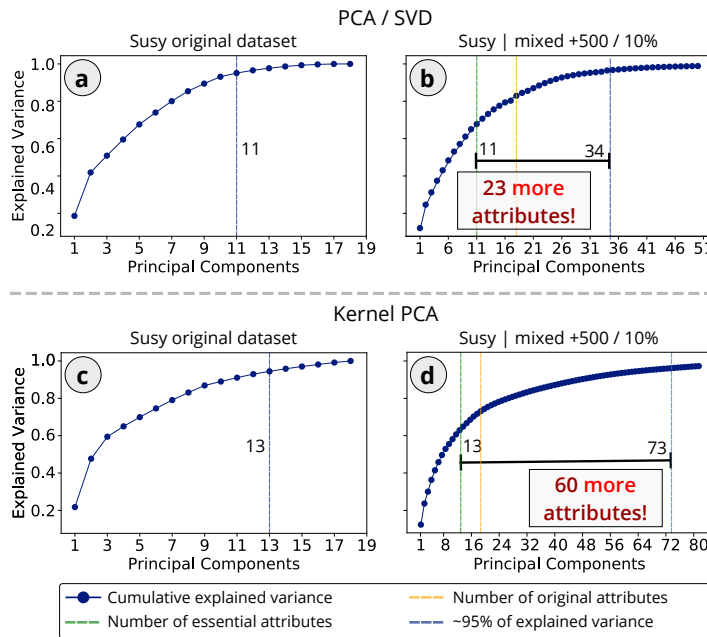
It is **important** to note that we evaluated the techniques PCA, SVD, PUFs, KPCA and FReE by performing all experiments described in Section 5.1 from Chapter 5 on **every one** of the 11 real datasets listed in Table 1. Similar patterns were observed in the results obtained from each dataset. Due to this fact, we report in the upcoming Sections 6.1.1 and 6.1.2 detailed results obtained from one of the datasets, i.e., Susy, and summarize for brevity in the upcoming Section 6.1.3 the results obtained from the other 10 datasets. Note, however, that the full results obtained from all 11 datasets are freely available for download in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

6.1.1 *Increasing the number of redundant attributes*

At first, we evaluated the ability of the techniques to detect several types of correlations with increasing numbers of redundant attributes, aimed at answering questions Q1, Q2 and Q4 from Section 5.1 of Chapter 5. To make it possible, let us explain how we quantified the accuracy of the variance-preservation techniques using dataset Susy as an example. Figure 13a reports the cumulative sum of the explained variance obtained after running PCA/SVD in the original dataset. Note that, according to PCA/SVD, to obtain a total explained variance of 95%, one needs

to use the first 11 principal components, *i.e.*, attributes, thus discarding the other 7 components as irrelevant ones. By contrast, under the same conditions, Figure 13c reports that KPCA suggested to maintain its first 13 principal components and to discard the other 5 components. By adding new redundant attributes in the dataset, we do not add extra information, and therefore the explained variance for each principal component should not be abruptly changed. With that in mind, when evaluating PCA, SVD and KPCA in any dataset with inserted redundant attributes, we quantified the estimated error as the number of additional components needed to explain 95% of the data variance, compared with the result obtained from the original dataset. In this sense, as the PUFs technique is based on the extraction of singular values through the SVD algorithm, and because it requires the user to inform the number of features that must be maintained, we also used as a basis the number of features necessary to maintain 95% of variance in the analyzed set, as well as PCA, SVD and KPCA. For example, Figure 13b reports that after inserting 500 new attributes in Susy with mixed correlations, PCA/SVD required 23 more components to explain 95% of the data variance, *i.e.*, $Error = 23$. Under the same conditions, KPCA’s error was 60; see Figure 13d.

Figure 13 – The cumulative explained variance of PCA, SVD and KPCA techniques. (a) and (b) respectively show the values obtained by PCA/SVD when applied on Susy original dataset, and Susy dataset with 500 additional redundant attributes formed by mixed correlations; (c) and (d) show the values obtained by KPCA under the same datasets. **Note:** similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.



Source: Elaborated by the author.

As it is described in Section 5.1.1 from Chapter 5, we added several redundant attributes to each of the 11 datasets studied, where each attribute is generated from one original attribute

employing a specific type of correlation with noise. Similar results were obtained from all datasets, so we detail in the following the results for one of them and summarize in the upcoming Section 6.1.3 the remaining results. Note, however, that the full results for all 11 datasets are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

Figure 14 reports the results obtained by PCA, SVD and KPCA from dataset Susy. The header $+m / \gamma\%$ represents the m redundant attributes added with γ percentage of noise used. D_{out} means the number of attributes that the techniques suggest must be maintained to explain at least 95% of the variance contained in the data, and Error means the absolute difference of the required quantity of attributes when compared with the result obtained from the original dataset. As it can be seen, for linear, logarithmic and sinusoidal correlations, PCA and SVD obtained relevant results. In the logarithmic and sinusoidal curves generated from the dataset, the techniques succeeded in obtaining an appropriate linear approximation, and consequently, they were able to detect such correlations. On the other hand, PCA and SVD presented poor results when applied to correlations of the types: quadratic, cubic, exponential, tenth power, hundredth power and mixed. For instance, when 500 new redundant attributes of mixed correlations were added with 5% of noise on the original data, to obtain a cumulative explained variance of 95%, as illustrated in Figure 13b, it would be necessary to use the first 34 principal components, more than 3 times what is required when compared with the result obtained from the original data.

In turn, KPCA obtained poor results for most cases; even for quadratic correlations, considering that its kernel is polynomial with a degree 2, KPCA performed much worse than PCA and SVD. For example, as it is illustrated in Figure 13d and Figure 14, using mixed correlations KPCA required up to 73 principal components to explain 95% of the data variance, which is more than 4 times the number of attributes contained in the original dataset.

Now, let us focus on the results of FReE and PUFS as they are feature selectors. To calculate their error, we obtained the absolute difference of the number of attributes selected as relevant from each one of the 11 datasets studied and its counterpart in a dataset with inserted redundant attributes. Additionally, because they are feature selectors, we verified the correctness of the attributes selected, *i.e.*, those that differ from the originally selected attributes and do not have any correlation with them are considered to be errors. The final Error of FReE and PUFS is, therefore, the number of attributes incorrectly selected plus the aforementioned absolute difference. D_{out} is the dimensionality of the final result.

For example, FReE automatically selected 7 relevant attributes from Susy while PUFS selected 11. When adding redundant attributes in the original data, this output must have a minimal change, since the new attributes do not add extra information. Figure 14 reports the results obtained by FReE and PUFS. Note that PUFS had quality results similar to those of PCA and SVD, while the output of our algorithm FReE had minimal variation, even after the insertion of up to 500 redundant attributes with noise, regardless of the type of correlation used. It demonstrates that our algorithm was able to correctly detect the existence of correlations. Let

Figure 14 – Experimental results of PCA, SVD, PUFs, KPCA and FReE when increasing the number of redundant attributes inserted in dataset Susy. **Note:** similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

PCA and SVD results																		
Correlation	+1 / 0.2%		+5 / 1.0%		+10 / 2.0%		+20 / 5.0%		+50 / 7.0%		+100 / 10.0%		+500 / 5.0%					
	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error				
Linear	11	0.0%	(0/1)	11	0.0%	(0/5)	11	0.0%	(0/10)	11	0.0%	(0/20)	11	0.0%	(0/100)	11	0.0%	(0/100)
Logarithmic	11	0.0%	(0/1)	11	0.0%	(0/5)	11	0.0%	(0/10)	11	0.0%	(0/20)	11	0.0%	(0/50)	11	0.0%	(0/100)
Quadratic	11	0.0%	(0/1)	13	40.0%	(2/5)	16	50.0%	(5/10)	17	30.0%	(6/20)	17	12.0%	(6/50)	17	6.0%	(6/100)
Cubic	12	100.0%	(1/1)	12	20.0%	(1/5)	14	30.0%	(3/10)	15	20.0%	(4/20)	17	12.0%	(6/50)	16	5.0%	(5/100)
Exponential	12	100.0%	(1/1)	14	60.0%	(3/5)	17	60.0%	(6/10)	18	35.0%	(7/20)	19	16.0%	(8/50)	17	6.0%	(6/100)
Tenth	12	100.0%	(1/1)	14	60.0%	(3/5)	18	70.0%	(7/10)	21	50.0%	(10/20)	23	24.0%	(12/50)	21	10.0%	(10/100)
Hundredth	12	100.0%	(1/1)	15	80.0%	(4/5)	15	40.0%	(4/10)	18	35.0%	(7/20)	29	36.0%	(18/50)	41	30.0%	(30/100)
Sinusoidal	11	0.0%	(0/1)	11	0.0%	(0/5)	11	0.0%	(0/10)	11	0.0%	(0/20)	11	0.0%	(0/50)	11	0.0%	(0/100)
Mixed	12	100.0%	(1/1)	14	60.0%	(3/5)	17	60.0%	(6/10)	20	45.0%	(9/20)	28	34.0%	(17/50)	27	16.0%	(16/100)

KPCA results																		
Correlation	+1 / 0.2%		+5 / 1.0%		+10 / 2.0%		+20 / 5.0%		+50 / 7.0%		+100 / 10.0%		+500 / 5.0%					
	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error				
Linear	14	100.0%	(1/1)	15	40.0%	(2/5)	16	30.0%	(3/10)	17	20.0%	(4/20)	19	12.0%	(6/50)	29	16.0%	(16/100)
Logarithmic	12	100.0%	(1/1)	12	20.0%	(1/5)	11	20.0%	(2/10)	11	10.0%	(2/50)	11	4.0%	(2/50)	11	0.0%	(2/100)
Quadratic	12	100.0%	(1/1)	17	80.0%	(4/5)	17	40.0%	(4/10)	21	40.0%	(8/20)	18	10.0%	(6/50)	30	17.0%	(17/100)
Cubic	13	0.0%	(0/1)	13	0.0%	(0/5)	16	30.0%	(3/10)	28	75.0%	(15/20)	38	50.0%	(25/50)	73	60.0%	(60/100)
Exponential	14	100.0%	(1/1)	14	20.0%	(1/5)	18	50.0%	(5/10)	25	60.0%	(12/20)	46	66.0%	(33/50)	37	24.0%	(24/100)
Tenth	12	100.0%	(1/1)	18	100.0%	(5/5)	12	10.0%	(1/10)	23	50.0%	(10/20)	38	50.0%	(25/50)	11	2.0%	(2/100)
Hundredth	14	100.0%	(1/1)	16	60.0%	(3/5)	18	50.0%	(5/10)	18	25.0%	(5/20)	28	30.0%	(15/50)	39	26.0%	(26/100)
Sinusoidal	12	100.0%	(1/1)	14	20.0%	(1/5)	15	20.0%	(2/10)	11	10.0%	(2/20)	11	4.0%	(2/50)	24	11.0%	(11/100)
Mixed	14	100.0%	(1/1)	15	40.0%	(2/5)	20	70.0%	(7/10)	22	45.0%	(9/20)	28	30.0%	(15/50)	60	47.0%	(47/100)

PUFs results																		
Correlation	+1 / 0.2%		+5 / 1.0%		+10 / 2.0%		+20 / 5.0%		+50 / 7.0%		+100 / 10.0%		+500 / 5.0%					
	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error				
Linear	11	0.0%	(0/1)	12	20.0%	(1/5)	12	10.0%	(1/10)	11	0.0%	(0/20)	11	0.0%	(0/50)	12	0.0%	(1/100)
Logarithmic	12	100.0%	(1/1)	12	20.0%	(1/5)	17	60.0%	(6/10)	11	0.0%	(0/20)	11	0.0%	(0/50)	12	0.0%	(1/100)
Quadratic	11	0.0%	(0/1)	14	60.0%	(3/5)	17	60.0%	(6/10)	19	40.0%	(8/20)	21	20.0%	(10/50)	21	6.0%	(10/100)
Cubic	13	200.0%	(1/1)	13	40.0%	(1/5)	15	40.0%	(3/10)	16	25.0%	(4/20)	18	14.0%	(6/50)	18	5.0%	(6/100)
Exponential	12	100.0%	(0/1)	15	80.0%	(3/5)	15	40.0%	(3/10)	19	40.0%	(7/20)	21	20.0%	(11/50)	20	6.0%	(10/100)
Tenth	13	200.0%	(1/1)	16	80.0%	(3/5)	18	70.0%	(6/10)	22	55.0%	(10/20)	25	28.0%	(13/50)	30	10.0%	(13/100)
Hundredth	12	100.0%	(1/1)	16	100.0%	(5/5)	16	50.0%	(5/10)	19	40.0%	(8/20)	32	42.0%	(31/50)	45	30.0%	(44/100)
Sinusoidal	12	100.0%	(0/1)	12	20.0%	(0/5)	12	10.0%	(0/10)	11	0.0%	(0/20)	11	0.0%	(1/50)	11	0.0%	(1/100)
Mixed	12	100.0%	(1/1)	14	60.0%	(3/5)	19	80.0%	(8/10)	22	55.0%	(11/20)	30	38.0%	(19/50)	28	16.0%	(17/100)

FReE results																		
Correlation	+1 / 0.2%		+5 / 1.0%		+10 / 2.0%		+20 / 5.0%		+50 / 7.0%		+100 / 10.0%		+500 / 5.0%					
	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error				
Linear	7	0.0%	(0/1)	7	0.0%	(0/5)	7	10.0%	(1/10)	7	5.0%	(1/20)	7	4.0%	(2/50)	8	3.0%	(3/100)
Logarithmic	7	0.0%	(0/1)	7	0.0%	(0/5)	7	0.0%	(0/10)	7	5.0%	(1/20)	7	2.0%	(1/50)	8	3.0%	(3/100)
Quadratic	7	0.0%	(0/1)	6	40.0%	(2/5)	8	20.0%	(2/10)	9	10.0%	(2/20)	10	6.0%	(3/50)	11	5.0%	(5/100)
Cubic	7	0.0%	(0/1)	7	60.0%	(3/5)	8	30.0%	(3/10)	8	15.0%	(3/20)	8	6.0%	(3/50)	9	3.0%	(3/100)
Exponential	7	0.0%	(0/1)	7	0.0%	(0/5)	7	0.0%	(0/10)	7	5.0%	(1/20)	7	6.0%	(3/50)	8	2.0%	(2/100)
Tenth	7	0.0%	(0/1)	7	0.0%	(0/5)	7	10.0%	(1/10)	7	0.0%	(0/20)	8	6.0%	(3/50)	8	4.0%	(4/100)
Hundredth	7	0.0%	(0/1)	7	0.0%	(0/5)	7	0.0%	(0/10)	7	0.0%	(0/20)	7	0.0%	(0/50)	7	0.0%	(0/100)
Sinusoidal	7	0.0%	(0/1)	7	0.0%	(0/5)	7	0.0%	(0/10)	7	5.0%	(1/20)	8	4.0%	(2/50)	8	3.0%	(3/100)
Mixed	6	100.0%	(1/1)	7	0.0%	(0/5)	8	20.0%	(2/10)	7	5.0%	(1/20)	8	4.0%	(2/50)	9	4.0%	(4/100)

Error	
Low	High
0	≥ 40

Source: Elaborated by the author.

us emphasize, as an example, the case illustrated in Figure 13, where PCA and SVD needed to use 23 more attributes, PUFs needed to use 35 more attributes and KPCA needed 60 more attributes to maintain 95% of the explained variance, while FReE incorrectly selected only 3 attributes among the total 518 attributes of the whole dataset. Similar results were obtained from the other 10 real datasets studied; see Figure 16 and Figure 17, and the repository mentioned at the beginning of Section 5.1 from Chapter 5.

6.1.2 Increasing the number of attributes per correlation

The previous section evaluated the ability of the techniques to remove redundant attributes formed by correlations that refer to only one original attribute each. Now, we focus on answering questions Q3 and Q4 from Section 5.1 of Chapter 5, by testing the ability of each technique to detect correlations referring to two or more original attributes.

Figure 15 – Experimental results of PCA, SVD, PUFs, KPCA and FReE when increasing the number of attributes per correlation in dataset Susy. **Note:** similar results were obtained from the other 10 real datasets studied; they are summarized in Figure 16 and Figure 17 for brevity, but the full results are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

PCA and SVD results															
Correlation	+2 / 1.0%			+6 / 2.0%			+10 / 5.0%			+14 / 7.0%			+18 / 10.0%		
	Dout		Error	Dout		Error	Dout		Error	Dout		Error	Dout		Error
Linear	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)
Logarithmic	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)
Quadratic	13	40.0%	(2/5)	13	40.0%	(2/5)	12	20.0%	(1/5)	12	20.0%	(1/5)	12	20.0%	(1/5)
Cubic	14	60.0%	(3/5)	13	40.0%	(2/5)	13	40.0%	(2/5)	13	40.0%	(2/5)	13	40.0%	(2/5)
Exponential	14	60.0%	(3/5)	12	20.0%	(1/5)	13	40.0%	(2/5)	13	40.0%	(2/5)	12	20.0%	(1/5)
Tenth	15	80.0%	(4/5)	12	20.0%	(1/5)	15	80.0%	(4/5)	12	20.0%	(1/5)	13	40.0%	(2/5)
Hundredth	15	80.0%	(4/5)	12	20.0%	(1/5)	12	20.0%	(1/5)	13	40.0%	(2/5)	14	60.0%	(3/5)
Sinusoidal	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)
Mixed	14	60.0%	(3/5)	15	80.0%	(4/5)	13	40.0%	(2/5)	14	60.0%	(3/5)	14	60.0%	(3/5)

KPCA results															
Correlation	+2 / 1.0%			+6 / 2.0%			+10 / 5.0%			+14 / 7.0%			+18 / 10.0%		
	Dout		Error	Dout		Error	Dout		Error	Dout		Error	Dout		Error
Linear	14	20.0%	(1/5)	13	0.0%	(0/5)	14	20.0%	(1/5)	13	0.0%	(0/5)	13	0.0%	(0/5)
Logarithmic	11	40.0%	(2/5)	11	40.0%	(2/5)	11	40.0%	(2/5)	11	40.0%	(2/5)	11	40.0%	(2/5)
Quadratic	14	20.0%	(1/5)	15	40.0%	(2/5)	15	40.0%	(2/5)	13	0.0%	(0/5)	14	20.0%	(1/5)
Cubic	12	20.0%	(1/5)	16	60.0%	(3/5)	12	20.0%	(1/5)	13	0.0%	(0/5)	13	0.0%	(0/5)
Exponential	14	20.0%	(1/5)	14	20.0%	(1/5)	14	20.0%	(1/5)	14	20.0%	(1/5)	14	20.0%	(1/5)
Tenth	19	120.0%	(6/5)	18	100.0%	(5/5)	13	0.0%	(0/5)	18	100.0%	(5/5)	17	80.0%	(4/5)
Hundredth	17	80.0%	(4/5)	15	40.0%	(2/5)	14	20.0%	(1/5)	14	20.0%	(1/5)	14	20.0%	(1/5)
Sinusoidal	14	20.0%	(1/5)	14	20.0%	(1/5)	13	0.0%	(0/5)	13	0.0%	(0/5)	13	0.0%	(0/5)
Mixed	16	60.0%	(3/5)	18	100.0%	(5/5)	16	60.0%	(3/5)	14	20.0%	(1/5)	16	60.0%	(3/5)

PUFS results															
Correlation	+2 / 1.0%			+6 / 2.0%			+10 / 5.0%			+14 / 7.0%			+18 / 10.0%		
	Dout		Error	Dout		Error	Dout		Error	Dout		Error	Dout		Error
Linear	12	20.0%	(1/5)	11	0.0%	(0/5)	12	20.0%	(1/5)	12	20.0%	(1/5)	12	20.0%	(1/5)
Logarithmic	12	20.0%	(1/5)	11	0.0%	(0/5)	12	20.0%	(1/5)	12	20.0%	(1/5)	11	0.0%	(0/5)
Quadratic	13	40.0%	(2/5)	14	60.0%	(3/5)	13	40.0%	(2/5)	12	20.0%	(1/5)	13	40.0%	(2/5)
Cubic	14	60.0%	(3/5)	14	60.0%	(3/5)	14	60.0%	(3/5)	14	60.0%	(3/5)	13	40.0%	(2/5)
Exponential	15	80.0%	(4/5)	13	40.0%	(2/5)	14	60.0%	(3/5)	14	60.0%	(3/5)	13	40.0%	(2/5)
Tenth	15	80.0%	(4/5)	12	20.0%	(1/5)	15	80.0%	(4/5)	13	40.0%	(2/5)	14	60.0%	(3/5)
Hundredth	15	80.0%	(4/5)	13	40.0%	(2/5)	13	40.0%	(2/5)	13	40.0%	(2/5)	14	60.0%	(3/5)
Sinusoidal	12	20.0%	(1/5)	12	20.0%	(1/5)	11	0.0%	(0/5)	11	0.0%	(0/5)	11	0.0%	(0/5)
Mixed	15	80.0%	(4/5)	17	120.0%	(6/5)	13	40.0%	(2/5)	14	60.0%	(3/5)	14	60.0%	(3/5)

FReE results															
Correlation	+2 / 1.0%			+6 / 2.0%			+10 / 5.0%			+14 / 7.0%			+18 / 10.0%		
	Dout		Error	Dout		Error	Dout		Error	Dout		Error	Dout		Error
Linear	7	0.0%	(0/5)	8	20.0%	(1/5)	8	40.0%	(2/5)	8	20.0%	(1/5)	9	60.0%	(3/5)
Logarithmic	6	20.0%	(1/5)	9	60.0%	(3/5)	8	60.0%	(3/5)	7	0.0%	(0/5)	8	40.0%	(2/5)
Quadratic	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)
Cubic	7	20.0%	(1/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)
Exponential	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)
Tenth	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)
Hundredth	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)
Sinusoidal	7	20.0%	(1/5)	7	20.0%	(1/5)	9	60.0%	(3/5)	8	20.0%	(1/5)	8	20.0%	(1/5)
Mixed	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)	7	0.0%	(0/5)

Error

Low 0 High 6

Source: Elaborated by the author.

Figure 15 details the results obtained from dataset Susy. The header $+n / \gamma\%$ represents that n original attributes chosen randomly were used to form each of the 5 new redundant attributes inserted in the dataset, considering $\gamma\%$ of noise. Similar to the previous experiments, in the case of PCA, SVD and KPCA, D_{out} means the number of principal components that need to be maintained to explain at least 95% of the variance. Error is the absolute difference between the number of components needed in the original dataset and its counterpart in a dataset with redundant attributes. For PUFs, D_{out} is the number of features that it needs to maintain to explain at least 95% of the variance; and for FReE, D_{out} is the number of attributes considered to be relevant in the dataset. For the feature selectors, Error is the number of attributes selected incorrectly, *i.e.*, those that differ from the originally selected attributes, and are not correlated with them, plus the absolute difference between the number of attributes selected in the original dataset and its counterpart in a dataset with redundancies.

Following the same behavior presented in the previous experiment, PCA, SVD and PUFS obtained good results when applied to the datasets that include linear, logarithmic and sinusoidal correlations, and yet obtained poor results in the datasets with quadratic, cubic, tenth power, hundredth power, exponential and mixed correlations. KPCA obtained poor results in most cases, even when using quadratic correlations. Observe that, for PCA, SVD, PUFS and KPCA, the number of attributes involved in each correlation did not have much relevance, since poor results were obtained even when the correlations involved only two original attributes each. On the other hand, our FReE obtained excellent results in most of the cases evaluated, with few exceptions. Even when applied to redundancies formed from 10 or more original attributes, the final quantity of dimensions had few changes, and FReE presented few errors by selecting barely the same attributes selected from the original dataset, thus reinforcing the stability and usability of the Fractal Theory for unsupervised dimensionality reduction tasks. Similar results were obtained from the other 10 real datasets studied; they are summarized in [Figure 16](#) and [Figure 17](#) for brevity, but the full results are freely available in the repository mentioned at the beginning of [Section 5.1](#) from [Chapter 5](#).


6.1.3 Summary of results from the other 10 real datasets

As it was described before, we performed the very same experiments on all of the 11 datasets studied and obtained similar results. For brevity, we detailed in the previous two sections, the results that refer to one of these datasets, i.e., Susy, and summarize in the following the results obtained from the other ones. The full results from all 11 datasets are in the repository mentioned at the beginning of [Section 5.1](#) from [Chapter 5](#). [Figure 16](#) and [Figure 17](#) report the results obtained with mixed correlations added to the other 10 datasets; we chose to report this particular set of results since it is the case that we consider to be the closest to the behavior of real-world data. Let us emphasize that KPCA had unfeasibly high runtime requirements to process datasets *Forex* and *Airline* with more than 5 redundant attributes inserted. Thus, we processed 10 uniform samples of 10% of the total points and reported the average of the results obtained. For PCA, SVD, PUFS and FReE, we still used the whole datasets.

[Figure 16](#) reports the results obtained when adding up to 500 redundant attributes in the datasets; each redundant attribute is correlated with one randomly-chosen original attribute. Note that the case with the highest dimensionality is when we add 500 new attributes to dataset *FMA*, thus leading to 1,018 attributes in total. As it can be seen, our proposed FReE consistently outperformed PCA, SVD, PUFS and KPCA in every single dataset. For example, when adding more than 20 redundant attributes in datasets *Hepmass*, *Airline* and *Higgs*, the variance-preservation techniques needed more components than the embedded dimensionality E of the original datasets, i.e., the total number of attributes; in some cases, they failed 24 times more than FReE. [Figure 17](#) reports the results obtained when increasing the number of attributes per correlation. As it was done with dataset *Susy*, we inserted 5 redundant attributes in each dataset

Figure 16 – Experimental results on increasing the number of redundant attributes for other 10 real-world datasets from different domains. **Note:** these results are presented in a summarized form for brevity, since they are similar to those obtained from dataset Susy that were already detailed. The full results regarding these 10 datasets are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

Technique	Datasets	+1 / 0.2%		+5 / 1.0%		+10 / 2.0%		+20 / 5.0%		+50 / 7.0%		+100 / 10.0%		+500 / 5.0%								
		Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error							
PCA / SVD	WeatherBR	12	0.0%	(0/1)	14	40.0%	(2/5)	16	40.0%	(4/10)	16	20.0%	(4/20)	18	12.0%	(6/50)	19	7.0%	(7/100)	24	2.4%	(12/500)
	Hepmass	23	100.0%	(1/1)	24	40.0%	(2/5)	25	30.0%	(3/10)	32	50.0%	(10/20)	34	24.0%	(12/50)	46	24.0%	(24/100)	56	6.8%	(34/500)
	Forex	8	200.0%	(2/1)	10	0.0%	(0/5)	10	0.0%	(0/10)	12	10.0%	(2/20)	18	16.0%	(8/50)	18	8.0%	(8/100)	21	2.2%	(11/500)
	NaturalGas	4	0.0%	(0/1)	5	20.0%	(1/5)	6	20.0%	(2/10)	4	0.0%	(0/20)	6	4.0%	(2/50)	5	1.0%	(1/100)	7	0.6%	(3/500)
	Airline	15	100.0%	(1/1)	15	20.0%	(1/5)	16	20.0%	(2/10)	20	30.0%	(6/20)	23	18.0%	(9/50)	26	12.0%	(12/100)	33	3.8%	(19/500)
	Household	10	0.0%	(0/1)	10	0.0%	(0/5)	10	0.0%	(0/10)	13	15.0%	(3/20)	12	4.0%	(2/50)	15	5.0%	(5/100)	18	1.6%	(8/500)
	Higgs	24	100.0%	(1/1)	25	40.0%	(2/5)	29	60.0%	(6/10)	30	35.0%	(7/20)	34	22.0%	(11/50)	45	22.0%	(22/100)	51	5.6%	(28/500)
	YearPredMSD	68	100.0%	(1/1)	69	40.0%	(2/5)	69	20.0%	(2/10)	70	15.0%	(3/20)	73	12.0%	(6/50)	76	9.0%	(9/100)	84	3.2%	(16/500)
	Fonts	41	100.0%	(1/1)	41	20.0%	(1/5)	43	30.0%	(3/10)	43	15.0%	(3/20)	45	10.0%	(5/50)	51	11.0%	(11/100)	53	2.6%	(13/500)
	FMA	81	100.0%	(1/1)	81	20.0%	(1/5)	82	20.0%	(2/10)	83	15.0%	(3/20)	84	8.0%	(4/50)	89	9.0%	(9/100)	91	2.2%	(11/500)
KPCA	WeatherBR	13	0.0%	(0/1)	15	40.0%	(2/5)	17	40.0%	(4/10)	17	20.0%	(4/20)	17	8.0%	(4/50)	21	8.0%	(8/100)	20	1.4%	(7/500)
	Hepmass	24	100.0%	(1/1)	26	60.0%	(3/5)	24	10.0%	(1/10)	36	65.0%	(13/20)	45	44.0%	(22/50)	53	30.0%	(30/100)	61	7.6%	(38/500)
	Forex	7	0.0%	(0/1)	10	60.0%	(3/5)	11	40.0%	(4/10)	13	30.0%	(6/20)	21	28.0%	(14/50)	27	20.0%	(20/100)	31	4.8%	(16/500)
	NaturalGas	4	100.0%	(1/1)	5	40.0%	(2/5)	6	30.0%	(3/10)	4	5.0%	(1/20)	5	4.0%	(2/50)	4	1.0%	(1/100)	4	0.2%	(1/500)
	Airline	17	10.0%	(1/1)	17	10.0%	(1/5)	19	30.0%	(3/10)	20	20.0%	(4/20)	25	18.0%	(9/50)	29	1.3	(13/100)	32	0.032	(16/500)
	Household	10	0.0%	(0/1)	10	0.0%	(0/5)	11	10.0%	(1/10)	14	20.0%	(4/20)	12	4.0%	(2/50)	16	6.0%	(6/100)	15	2.0%	(10/500)
	Higgs	24	100.0%	(1/1)	22	20.0%	(1/5)	29	60.0%	(6/10)	34	55.0%	(11/20)	36	26.0%	(13/50)	46	23.0%	(23/100)	49	5.2%	(26/500)
	YearPredMSD	75	100.0%	(1/1)	75	20.0%	(1/5)	77	30.0%	(3/10)	80	30.0%	(6/20)	83	18.0%	(9/50)	87	13.0%	(13/100)	92	3.6%	(18/500)
	Fonts	44	100.0%	(1/1)	45	40.0%	(2/5)	47	40.0%	(4/10)	49	30.0%	(6/20)	52	18.0%	(9/50)	58	15.0%	(15/100)	62	3.8%	(19/500)
	FMA	83	100.0%	(1/1)	83	20.0%	(1/5)	85	30.0%	(3/10)	85	15.0%	(3/20)	87	10.0%	(5/50)	93	11.0%	(11/100)	94	2.4%	(12/500)
PUFS	WeatherBR	13	100.0%	(1/1)	15	60.0%	(3/5)	17	50.0%	(5/10)	17	25.0%	(5/20)	19	14.0%	(7/50)	20	8.0%	(8/100)	24	2.4%	(12/500)
	Hepmass	23	100.0%	(1/1)	26	80.0%	(4/5)	26	40.0%	(4/10)	34	60.0%	(12/20)	37	20.0%	(15/50)	49	27.0%	(27/100)	60	7.6%	(38/500)
	Forex	8	0.0%	(0/1)	11	100.0%	(5/5)	10	40.0%	(4/10)	13	30.0%	(6/20)	20	28.0%	(14/50)	19	13.0%	(13/100)	22	3.2%	(16/500)
	NaturalGas	4	0.0%	(0/1)	5	20.0%	(1/5)	7	30.0%	(3/10)	4	10.0%	(2/20)	6	8.0%	(4/50)	5	3.0%	(3/100)	8	1.2%	(6/500)
	Airline	15	100.0%	(1/1)	17	60.0%	(3/5)	17	30.0%	(3/10)	21	35.0%	(7/20)	25	22.0%	(11/50)	28	14.0%	(14/100)	36	4.4%	(22/500)
	Household	10	0.0%	(0/1)	11	20.0%	(1/5)	10	0.0%	(0/10)	13	15.0%	(3/20)	13	6.0%	(3/50)	17	7.0%	(7/100)	19	1.8%	(9/500)
	Higgs	26	100.0%	(1/1)	27	40.0%	(2/5)	31	60.0%	(6/10)	32	35.0%	(7/20)	36	22.0%	(11/50)	49	24.0%	(24/100)	51	5.2%	(26/500)
	YearPredMSD	68	100.0%	(1/1)	73	120.0%	(6/5)	75	80.0%	(8/10)	75	40.0%	(8/20)	74	14.0%	(7/50)	76	9.0%	(9/100)	91	4.8%	(24/500)
	Fonts	43	100.0%	(1/1)	43	20.0%	(1/5)	44	20.0%	(2/10)	46	20.0%	(4/20)	49	14.0%	(7/50)	51	9.0%	(9/100)	55	2.6%	(13/500)
	FMA	81	100.0%	(1/1)	81	20.0%	(1/5)	84	40.0%	(4/10)	85	25.0%	(5/20)	92	20.0%	(10/50)	90	8.0%	(8/100)	96	2.8%	(14/500)
FRrE	WeatherBR	8	0.0%	(0/1)	7	40.0%	(2/5)	7	10.0%	(1/10)	8	5.0%	(1/20)	5	8.0%	(4/50)	5	3.0%	(3/100)	5	0.6%	(3/500)
	Hepmass	14	0.0%	(0/1)	14	0.0%	(0/5)	14	0.0%	(0/10)	15	5.0%	(1/20)	14	4.0%	(2/50)	16	2.0%	(2/100)	14	0.6%	(3/500)
	Forex	8	0.0%	(0/1)	9	20.0%	(1/5)	9	40.0%	(4/10)	9	5.0%	(1/20)	12	14.0%	(7/50)	12	6.0%	(6/100)	11	1.0%	(5/500)
	NaturalGas	2	100.0%	(1/1)	2	20.0%	(1/5)	2	10.0%	(1/10)	2	5.0%	(1/20)	2	2.0%	(1/50)	2	2.0%	(2/100)	2	0.4%	(2/500)
	Airline	6	100.0%	(1/1)	6	20.0%	(1/5)	6	20.0%	(2/10)	6	5.0%	(1/20)	6	5.0%	(1/50)	6	5.0%	(1/100)	6	0.2%	(1/500)
	Household	6	0.0%	(0/1)	6	20.0%	(1/5)	6	0.0%	(0/10)	6	5.0%	(1/20)	5	6.0%	(3/50)	6	2.0%	(2/100)	6	0.4%	(2/500)
	Higgs	9	100.0%	(1/1)	9	0.0%	(0/5)	9	10.0%	(1/10)	9	5.0%	(1/20)	9	2.0%	(1/50)	10	4.0%	(4/100)	9	0.2%	(1/500)
	YearPredMSD	7	100.0%	(1/1)	4	80.0%	(4/5)	4	40.0%	(4/10)	6	15.0%	(3/20)	7	4.0%	(2/50)	5	4.0%	(4/100)	6	0.8%	(3/500)
	Fonts	4	0.0%	(0/1)	3	20.0%	(1/5)	5	10.0%	(1/10)	4	0.0%	(0/20)	4	2.0%	(1/50)	3	1.0%	(1/100)	4	0.2%	(1/500)
	FMA	7	200.0%	(2/1)	7	40.0%	(2/5)	4	10.0%	(1/10)	7	10.0%	(2/20)	7	6.0%	(3/50)	8	4.0%	(4/100)	4	0.4%	(2/500)

Low (0)  High (≥ 30)

Source: Elaborated by the author.

in such a way that each new attribute is correlated with two or more original ones. The maximum number of original attributes per correlation is the embedded dimensionality E of each dataset, or it is 50 when $E > 50$. As it can be seen, our algorithm FRrE accurately removed almost all redundancies, consistently outperforming PCA, SVD, PUFS and KPCA, which again obtained poor results in many cases, even when each redundant attribute was formed from only 2 original ones. Considering the aforementioned results, we conclude that the patterns reported in this chapter may represent a variety of real-world scenarios since they were observed in 11 datasets from distinct domains.

6.1.4 Scalability comparison between the approaches studied

Our proposed FRrE method showed to be effective in removing redundant attributes from all 11 datasets studied, detecting both linear and non-linear correlations, unlike the compared methods that are based on variance preservation, PCA, SVD, PUFS and KPCA.

However, in addition to result quality experiments, we decided to perform a comparative analysis between the complexity of the algorithms and to present such nature in scalability

Figure 17 – Experimental results on increasing the number of original attributes per correlation for other 10 real-world datasets from different domains. **Note:** these results are presented in a summarized form for brevity, as they are similar to those obtained from dataset Susy that were already detailed. The full results for these 10 real datasets are freely available in the repository mentioned at the beginning of Section 5.1 from Chapter 5.

Technique	Datasets	+2 / 0.2%		+4 / 1.0%		+8 / 2.0%		+12 / 5.0%		+20 / 7.0%		+28 / 10.0%		+50 / 10.0%	
		Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error	Dout	Error
PCA / SVD	WeatherBR	13	20.0% (1/5)	14	40.0% (2/5)	15	60.0% (3/5)	15	60.0% (3/5)	13	20.0% (1/5)	-	-	-	-
	Hepmass	26	80.0% (4/5)	26	80.0% (4/5)	26	80.0% (4/5)	26	80.0% (4/5)	24	40.0% (2/5)	25	60.0% (3/5)	-	-
	Forex	9	20.0% (1/5)	10	0.0% (0/5)	10	0.0% (0/5)	10	0.0% (0/5)	-	-	-	-	-	-
	NaturalGas	5	40.0% (1/5)	5	40.0% (1/5)	-	-	-	-	-	-	-	-	-	-
	Airline	18	80.0% (4/5)	18	80.0% (4/5)	18	80.0% (4/5)	18	80.0% (4/5)	16	40.0% (2/5)	-	-	-	-
	Household	10	0.0% (0/5)	12	40.0% (2/5)	11	20.0% (1/5)	10	0.0% (0/5)	-	-	-	-	-	-
	Higgs	24	20.0% (1/5)	25	40.0% (2/5)	26	60.0% (3/5)	27	80.0% (4/5)	27	80.0% (4/5)	26	60.0% (3/5)	-	-
	YearPredMSD	68	20.0% (1/5)	69	40.0% (2/5)	70	60.0% (3/5)	70	60.0% (3/5)	71	80.0% (4/5)	71	80.0% (4/5)	71	80.0% (4/5)
	Fonts	41	20.0% (1/5)	42	40.0% (2/5)	43	60.0% (3/5)	44	80.0% (4/5)	43	60.0% (3/5)	44	80.0% (4/5)	44	80.0% (4/5)
	FMA	81	20.0% (1/5)	83	60.0% (3/5)	83	60.0% (3/5)	83	60.0% (3/5)	83	60.0% (3/5)	84	80.0% (4/5)	84	80.0% (4/5)
KPCA	WeatherBR	12	20.0% (1/5)	14	20.0% (1/5)	14	20.0% (1/5)	16	40.0% (2/5)	14	20.0% (1/5)	-	-	-	-
	Hepmass	21	40.0% (2/5)	24	20.0% (1/5)	22	20.0% (1/5)	19	80.0% (4/5)	15	240.0% (8/5)	11	240.0% (12/5)	-	-
	Forex	14	140.0% (7/5)	14	140.0% (7/5)	13	120.0% (6/5)	15	160.0% (8/5)	-	-	-	-	-	-
	NaturalGas	5	40.0% (2/5)	5	40.0% (2/5)	-	-	-	-	-	-	-	-	-	-
	Airline	19	60.0% (3/5)	19	60.0% (3/5)	18	40.0% (2/5)	19	60.0% (3/5)	19	60.0% (3/5)	-	-	-	-
	Household	11	20.0% (1/5)	12	40.0% (2/5)	12	40.0% (2/5)	10	0.0% (0/5)	-	-	-	-	-	-
	Higgs	25	40.0% (2/5)	24	20.0% (1/5)	25	40.0% (2/5)	28	100.0% (5/5)	22	20.0% (1/5)	27	80.0% (4/5)	-	-
	YearPredMSD	76	40.0% (2/5)	76	40.0% (2/5)	77	60.0% (3/5)	76	40.0% (2/5)	78	80.0% (4/5)	78	80.0% (4/5)	79	100.0% (5/5)
	Fonts	44	40.0% (1/5)	45	40.0% (2/5)	45	40.0% (2/5)	46	60.0% (3/5)	47	80.0% (4/5)	48	100.0% (5/5)	48	100.0% (5/5)
	FMA	84	60.0% (2/5)	85	60.0% (3/5)	85	60.0% (3/5)	86	80.0% (4/5)	86	80.0% (4/5)	88	120.0% (6/5)	87	100.0% (5/5)
PUFS	WeatherBR	14	40.0% (2/5)	15	40.0% (3/5)	15	60.0% (3/5)	16	60.0% (3/5)	14	20.0% (1/5)	-	-	-	-
	Hepmass	27	100.0% (5/5)	27	80.0% (5/5)	29	80.0% (7/5)	28	80.0% (4/5)	25	40.0% (2/5)	26	60.0% (3/5)	-	-
	Forex	10	40.0% (2/5)	11	0.0% (3/5)	10	0.0% (2/5)	11	0.0% (3/5)	-	-	-	-	-	-
	NaturalGas	5	20.0% (1/5)	5	20.0% (1/5)	-	-	-	-	-	-	-	-	-	-
	Airline	18	80.0% (4/5)	19	80.0% (5/5)	18	80.0% (4/5)	20	80.0% (6/5)	17	40.0% (3/5)	-	-	-	-
	Household	10	0.0% (3/5)	13	40.0% (3/5)	12	20.0% (2/5)	11	0.0% (1/5)	-	-	-	-	-	-
	Higgs	26	60.0% (3/5)	25	40.0% (2/5)	28	60.0% (5/5)	29	80.0% (6/5)	29	80.0% (6/5)	27	60.0% (4/5)	-	-
	YearPredMSD	68	20.0% (1/5)	75	40.0% (7/5)	77	60.0% (9/5)	74	60.0% (6/5)	71	80.0% (3/5)	75	80.0% (5/5)	74	80.0% (6/5)
	Fonts	45	100.0% (5/5)	43	40.0% (3/5)	45	60.0% (3/5)	47	80.0% (5/5)	46	60.0% (4/5)	48	80.0% (6/5)	44	80.0% (2/5)
	FMA	82	40.0% (2/5)	90	60.0% (10/5)	85	60.0% (5/5)	90	60.0% (10/5)	85	60.0% (5/5)	90	80.0% (10/5)	89	80.0% (9/5)
FRFE	WeatherBR	8	0.0% (0/5)	9	20.0% (2/5)	7	40.0% (2/5)	7	40.0% (2/5)	7	40.0% (2/5)	-	-	-	-
	Hepmass	13	20.0% (1/5)	14	0.0% (0/5)	14	0.0% (0/5)	14	0.0% (0/5)	13	20.0% (1/5)	13	20.0% (1/5)	-	-
	Forex	8	0.0% (0/5)	8	0.0% (0/5)	8	0.0% (0/5)	8	0.0% (0/5)	-	-	-	-	-	-
	NaturalGas	2	40.0% (2/5)	2	20.0% (1/5)	-	-	-	-	-	-	-	-	-	-
	Airline	6	20.0% (1/5)	6	0.0% (0/5)	6	0.0% (0/5)	6	0.0% (0/5)	6	0.0% (0/5)	-	-	-	-
	Household	5	20.0% (1/5)	5	20.0% (1/5)	6	20.0% (1/5)	7	20.0% (1/5)	-	-	-	-	-	-
	Higgs	9	40.0% (2/5)	9	40.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)
	YearPredMSD	7	0.2 (1/5)	8	0.0% (0/5)	8	0.0% (0/5)	9	20.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)	9	20.0% (1/5)
	Fonts	4	0.0% (0/5)	4	0.0% (0/5)	4	0.0% (0/5)	4	0.0% (0/5)	4	0.0% (0/5)	5	20.0% (1/5)	5	20.0% (1/5)
	FMA	6	20.0% (1/5)	6	20.0% (1/5)	7	60.0% (2/5)	6	20.0% (1/5)	6	20.0% (1/5)	6	20.0% (1/5)	6	20.0% (1/5)

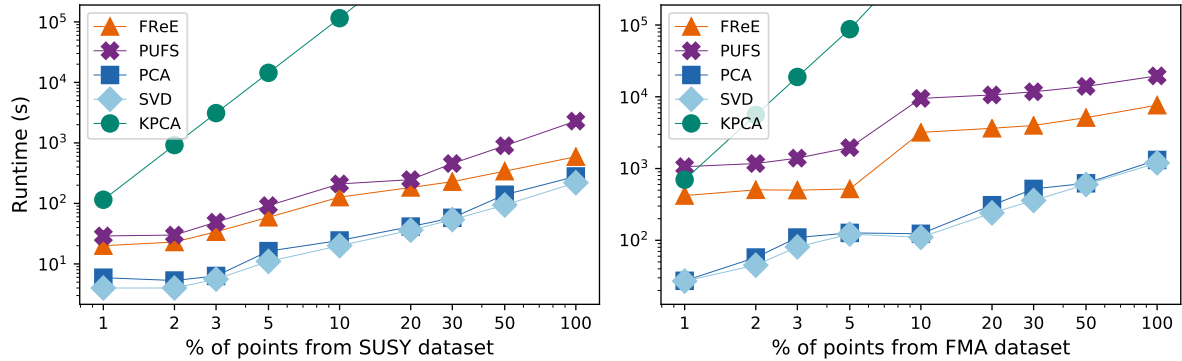
Low (0)  High (≥ 6)

Source: Elaborated by the author.

experiments by increasing the number of data points. Figure 18 reports runtime results in seconds obtained by processing random samples with increasing sizes from Susy and from the dataset with the highest dimensionality, i.e., FMA, up to the full datasets. The results are the average over ten distinct runs; standard deviation values are too small to be shown; besides that, note that the plot's axes are in log-scale.

As it was previously described in this monograph, PCA and SVD are algorithms popularly used to support several data analysis tasks, one of which is the dimensionality reduction. Algorithm PCA has two computationally intensive steps, which is the computation of the covariance matrix, and the eigenvalues decomposition of the covariance matrix. This fact results in a complexity $O(N \times d \times \min(N, d) + d^3)$, where N is the number of points in a dataset and d is the number of features. Whereas algorithm SVD obtains the same quality results, but it differs by making the step of singular values decomposition from a mean-centered matrix, which leaves the complexity of the algorithm to $O(N \times d^2 + d^3)$. In general, when dealing with Big Data, the number of points in the dataset is much higher than the number of dimensions. In Figure 18, the results of the scalability experiments regarding the number of points show that

Figure 18 – Runtime of PCA, SVD, PUFS, KPCA and FReE in seconds when selecting features from random samples of Susy and FMA (highest dimensionality), up to the full datasets. Our proposed FReE scales linearly on the data size.



Source: Elaborated by the author.

PCA and SVD reveal this linear behavior for the number of points. Algorithm KPCA, in turn, is capable of detecting non-linear correlations from the kernel functions used. However, this algorithm has some issues about computational complexity, which limits its use in practical applications. KPCA has a major bottleneck when it needs to compute the eigenvector of a $N \times N$ kernel matrix, where N is the number of points in the dataset analyzed. This computation results in a complexity of $O(N^2)$ for storing the kernel matrix and $O(N^3)$ in processing time. As it is shown in Figure 18, KPCA had the worst performance among the analyzed methods, being able to process only a small part of the datasets within a limit of up to 48 hours of processing.

PUFS is a feature selector that works in an unsupervised and distributed way; it is a technique based on data variance, using the SVD as a basis, being limited to detecting only linear correlations. The complexity of the PUFS is $O(N \times d \times \min(N, d))$ to compute the singular vectors, and $O(d^3 \times \log(d))$ for the deterministic phase in which the feature set is reorganized in order of importance. Consequently, it has cubic complexity regarding the number of features in the dataset. Figure 18 illustrates its scalability results, showing that PUFS has better performance when compared to KPCA, but it is slower than the other algorithms, PCA, SVD and FReE.

In turn, our proposed algorithm FReE has the complexity described in Section 4.1 from Chapter 4. It presents linear scalability regarding the data cardinality. In Figure 18, it shows worse results when compared to the PCA and SVD algorithms, which detect only linear correlations, but it was shown to be faster than the PUFS and KPCA algorithms which also seek to find non-linear correlations.

6.1.5 Final Considerations

This chapter presented the results obtained from the comparison between the unsupervised dimensionality reduction algorithms based on variance preservation and our newly

proposed technique that is based on the Fractal Theory. The results show that the fractal-based technique, FReE, was able to accurately detect the various types of correlations, unlike algorithms PCA, SVD and PUFS, or even algorithm KPCA that proposes to detect non-linear correlations. Also, our new algorithm proved to be scalable on the number of points in the dataset, being faster than algorithms PUFS and KPCA. The next chapter presents the conclusions of this MSc work.

CONCLUSION

This MSc work investigated the following hypothesis: “the application of concepts from the Fractal Theory in dimensionality reduction tasks, using massively parallel processing via Apache Spark, and maximizing the use of main memory instead of secondary memory allows the development of an effective and efficient technique that is capable of reducing the dimensionality of Terabytes or even Petabytes of complex data”. We corroborated it by performing an extensive exploratory study through 11 real-world datasets from multiple domains, and comparing two distinct approaches for unsupervised dimensionality reduction: (a) the standard variance preservation, and; (b) one alternative, fractal-based solution for which we proposed one new, fast and scalable Spark-based algorithm named FReE. Our main contributions are presented in the following.

7.1 Main Contributions of this MSc Work

The main contributions accomplished in this MSc work are:

C1 – Extensive exploratory evaluation: We reported results of a detailed exploratory study, using 11 datasets with up to 123.5 million elements and 518 attributes from physics, finance, transportation, energy, electricity, image, audio and climatic domains, systematically evaluating and validating the ability of the variance preservation and the fractal-based approaches to remove many types of attribute correlations. To the best of our knowledge, this is the first work to present a comparative study between variance-preservation techniques and those that are based on the Fractal Theory, by systematically exploring their limitations to remove different correlation types under a variety of circumstances;

C2 – Novel algorithm: We proposed the new algorithm Fractal Redundancy Elimination – FReE, a parallel and distributed dimensionality reduction algorithm that uses concepts from the Fractal Theory and Apache Spark to deal with data of high cardinality. FReE also

implements a novel feature partitioning strategy that we carefully developed to make it suited for high-dimensionality data processing. To the best of our knowledge, this is the first fractal-based algorithm that is capable of processing billion-scale-elements datasets with hundreds or even thousands of attributes.

7.2 Discussion and Future Work

We showed that, at least for large datasets of dimensionality with up to $\sim 1,000$ attributes, our new fractal-based algorithm FReE is the best option, being scalable to process millions or even billions of objects and more accurate to eliminate correlations of many kinds, such as linear, quadratic, logarithmic and exponential, even when there are up to 500 redundant attributes in a dataset, or correlations that depend on up to dozens of the original attributes. Indeed, it is also important to emphasize that our proposal has some limitations. Its first requirement is to process large volumes of data, since the number of objects required to enable the analysis of a dataset increases together with its intrinsic dimensionality. Fortunately, large datasets abound nowadays. As a consequence, we point out that the increasing volume of data available in the current era of Big Data enables the use of the Fractal Theory to spot relevant attributes in data of high dimensionality. Besides, it is also required that the datasets to be processed exhibit self-similarity properties. Fortunately, it is well-known in the literature that most real-world datasets present statistical self-similarity, such as the 11 datasets from distinct domains that we investigated.

With regard to future work, the contributions resulting from this MSc project support some studies on dimensionality reduction for Big Data. As an example, one in-depth evaluation of strategies for partitioning the input dataset could be performed, aiming to assess the relevance of partitions and their selected features. In this work, we explored an approach that uses the original sequence of the set of features to partition the data into blocks of fixed size. Other strategies can be studied to improve the performance of the algorithm in detecting correlations in each block, possibly including the use of feature reordering and blocks with distinct sizes. Besides that, as it was detailed throughout this monograph, we explored the context of dimensionality reduction in an unsupervised way, so to support activities such as clustering and outlier detection. For the supervised context, an interesting idea would be to use the Theory of Fractals in regression problems, so to maximize the information gain of independent variables concerning the dependent variable. That is, in addition of using the Fractal Theory to remove linear and non-linear correlations from the dataset, one could use the variable information that is provided for the training set to maximize the importance of the non-redundant features of the set.

Finally, let us highlight that this MSc work generated the following publications:

1. **P1** – Jadson J. M. Oliveira, Robson L. F. Cordeiro: Unsupervised Dimensionality Reduction: are we going to the right direction?. In: Knowledge-based Systems, Elsevier, 196:105777, 2020, 14 pages, DOI:10.1016/j.knosys.2020.105777, **Journal Qualis A1**;

2. **P2** – Lucas F. Kunze, Thábata Amaral, Leonardo M. P. Morais, Jadson J. M. Oliveira, Altamir G. Bispo Junior, Elaine P. M. de Sousa, Robson L. F. Cordeiro: Classification Analysis of NDVI Time Series in Metric Spaces for Sugarcane Identification. In: 20th International Conference on Enterprise Information Systems — ICEIS, 2018, Funchal, Portugal. p. 162-169, DOI:10.5220/0006709401620169, **International Conference Qualis A3**.

BIBLIOGRAPHY

ALSHEIKH, M. A.; NIYATO, D.; LIN, S.; TAN, H.-P.; HAN, Z. Mobile big data analytics using deep learning and apache spark. **IEEE network**, IEEE, v. 30, n. 3, p. 22–29, 2016. Citation on page [41](#).

ASFOOR, H.; SRINIVASAN, R.; VASUDEVAN, G.; VERBIEST, N.; CORNELLS, C.; TOLENTINO, M.; TEREDESAL, A.; COCK, M. D. Computing fuzzy rough approximations in large scale information systems. In: IEEE. **Big Data (Big Data), 2014 IEEE International Conference on**. [S.l.], 2014. p. 9–16. Citation on page [38](#).

BAIOCO, G. B.; TRAINA, A. J. M.; TRAINA JR, C. Mamcost: Global and local estimates leading to robust cost estimation of similarity queries. In: **SSDBM**. [S.l.: s.n.], 2007. p. 6–16. Citation on page [36](#).

BALCAN, M. F.; LIANG, Y.; SONG, L.; WOODRUFF, D.; XIE, B. Communication efficient distributed kernel principal component analysis. In: **KDD**. [S.l.: s.n.], 2016. p. 725–734. Citations on pages [32](#), [33](#), [34](#), [45](#), and [61](#).

BARBARÁ, D.; CHEN, P. Using the fractal dimension to cluster datasets. In: **ACM SIGKDD**. Boston, MA: [s.n.], 2000. p. 260–264. Citation on page [36](#).

BONES, C. C.; ROMANI, L. A. S.; SOUSA, E. P. M. de. Improving multivariate data streams clustering. **Procedia Computer Science**, v. 80, p. 461–471, 2016. ICCS. Citation on page [36](#).

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001. Citation on page [48](#).

BRYANT, A.; CIOS, K. Rnn-dbscan: A density-based clustering algorithm using reverse nearest neighbor density estimates. **IEEE Transactions on Knowledge and Data Engineering**, v. 30, n. 6, p. 1109–1121, 2018. Citation on page [31](#).

BöHM, C. A cost model for query processing in high dimensional data spaces. **ACM TODS**, v. 25, n. 2, p. 129–178, 2000. Citation on page [36](#).

CHAKRABARTI, D.; FALOUTSOS, C. F4: large-scale automated forecasting using fractals. In: **CIKM**. [S.l.: s.n.], 2002. v. 1, p. 2–9. Citation on page [36](#).

CHEN, H.; LI, T.; CAI, Y.; LUO, C.; FUJITA, H. Parallel attribute reduction in dominance-based neighborhood rough set. **Inf. Sci.**, v. 373, p. 351–368, 2016. ISSN 0020-0255. Citation on page [37](#).

CHENG, K.; LI, J.; LIU, H. Unsupervised feature selection in signed social networks. In: **KDD**. [S.l.: s.n.], 2017. p. 777–786. Citations on pages [26](#) and [31](#).

CLARK, J.; PROVOST, F. Unsupervised dimensionality reduction versus supervised regularization for classification from sparse data. **Data Mining and Knowledge Discovery**, v. 33, n. 4, p. 871–916, 2019. Citation on page [30](#).

CORDEIRO, R. L. F.; FALOUTSOS, C.; TRAINA JR, C. **Data mining in large sets of complex data**. [S.l.]: Springer-Verlag New York Incorporated, 2013. Citations on pages 25, 29, 30, and 39.

CORDEIRO, R. L. F.; TRAINA, A. J. M.; FALOUTSOS, C.; TRAINA JR, C. Halite: fast and scalable multiresolution local-correlation clustering. **IEEE TKDE**, v. 25, n. 2, p. 387–401, 2013. Citations on pages 31 and 36.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107–113, 2008. Citation on page 38.

DING, Y.; ZHU, G.; CUI, C.; ZHOU, J.; TAO, L. A parallel implementation of singular value decomposition based on map-reduce and parpack. In: **International Conference on Computer Science and Network Technology**. [S.l.: s.n.], 2011. v. 2, p. 739–741. Citations on pages 33 and 44.

DOBRE, C.; XHAFI, F. Intelligent services for big data science. **Future Generation Computer Systems**, v. 37, p. 267 – 281, 2014. ISSN 0167-739X. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications. Available: <<http://www.sciencedirect.com/science/article/pii/S0167739X13001593>>. Citation on page 25.

DU, T. Y. Dimensionality reduction techniques for visualizing morphometric data: Comparing principal component analysis to nonlinear methods. **Evolutionary Biology**, 2018. Citation on page 48.

ELGAMAL, T.; YABANDEH, M.; ABOULNAGA, A.; MUSTAFA, W.; HEFEEDA, M. spca: Scalable principal component analysis for big data on distributed platforms. In: **SIGMOD**. [S.l.: s.n.], 2015. p. 79–91. Citations on pages 32, 33, and 44.

ERL, T.; PUTTINI, R.; MAHMOOD, Z. **Cloud computing: concepts, technology & architecture**. [S.l.]: Pearson Education, 2013. Citation on page 37.

FALOUTSOS, C.; KAMEL, I. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In: **PODS**. [S.l.]: ACM, 1994. p. 4–13. Citations on pages 25, 31, and 34.

FALOUTSOS, C.; SEEGER, B.; TRAINA, A. J. M.; TRAINA JR, C. Spatial join selectivity using power laws. In: **SIGMOD**. [S.l.: s.n.], 2000. p. 177–188. Citation on page 36.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. Advances in knowledge discovery and data mining. In: FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. (Ed.). Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. chap. From Data Mining to Knowledge Discovery: An Overview, p. 1–34. ISBN 0-262-56097-6. Available: <<http://dl.acm.org/citation.cfm?id=257938.257942>>. Citations on pages 29, 30, and 31.

FOURNIER, Q.; ALOISE, D. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In: **2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)**. [S.l.: s.n.], 2019. p. 211–214. Citations on pages 36 and 37.

FRAIDEINBERZE, A. C.; RODRIGUES, J. F.; CORDEIRO, R. L. F. Effective and unsupervised fractal-based feature selection for very large datasets: Removing linear and non-linear attribute correlations. In: **ICDMW**. [S.l.: s.n.], 2016. p. 615–622. ISSN 2375-9259. Citations on pages [25](#), [31](#), [32](#), [33](#), [34](#), [35](#), [36](#), [45](#), [46](#), and [56](#).

GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. **IDC iView: IDC Analyze the future**, v. 2007, n. 2012, p. 1–16, 2012. Citation on page [25](#).

GISBRECHT, A.; HAMMER, B. Data visualization by nonlinear dimensionality reduction. **Wiley Int. Rev. Data Min. and Knowl. Disc.**, v. 5, n. 2, p. 51–73, 2015. Citation on page [48](#).

GOLAY, J.; KANEVSKI, M. Unsupervised feature selection based on the morisita estimator of intrinsic dimension. **Knowledge-Based Systems**, v. 135, p. 125–134, 2017. ISSN 0950-7051. Citations on pages [25](#), [26](#), [31](#), [36](#), [45](#), and [48](#).

GU, L.; LI, H. Memory or time: Performance evaluation for iterative operation on hadoop and spark. In: IEEE. **High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on**. [S.l.], 2013. p. 721–727. Citations on pages [40](#) and [41](#).

GUILLÉN, A.; ARENAS, M. I. G.; HEESWIJK, M. van; SOVILJ, D.; LENDASSE, A.; HERRERA, L. J.; POMARES, H.; ROJAS, I. Fast feature selection in a gpu cluster using the delta test. **Entropy**, Multidisciplinary Digital Publishing Institute, v. 16, n. 2, p. 854–869, 2014. Citation on page [38](#).

HALKO, N. P. Randomized methods for computing low-rank approximations of matrices. 2012. Citation on page [44](#).

HAN, J.; PEI, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. [S.l.]: Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780123814807. Citations on pages [30](#) and [31](#).

_____. **Data Mining: Concepts and Techniques**. [S.l.]: Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). Citation on page [33](#).

HAUSER, R. A.; EFTEKHARI, A.; MATZINGER, H. F. Pca by determinant optimisation has no spurious local optima. In: **KDD**. [S.l.: s.n.], 2018. p. 1504–1511. Citations on pages [33](#) and [44](#).

KARAU, H.; KONWINSKI, A.; WENDELL, P.; ZAHARIA, M. **Learning spark: lightning-fast big data analysis**. [S.l.]: " O'Reilly Media, Inc.", 2015. Citation on page [40](#).

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. **Mining of Massive Datasets**. [S.l.]: Cambridge University Press, 2014. ISBN 9781107077232. Citations on pages [30](#) and [31](#).

LI, J.; CHENG, K.; WANG, S.; MORSTATTER, F.; TREVINO, R. P.; TANG, J.; LIU, H. Feature selection: A data perspective. **ACM Computing Surveys**, v. 50, n. 6, p. 94:1–94:45, 2017. Citations on pages [32](#) and [37](#).

LI, J.; LIU, H. Challenges of feature selection for big data analytics. **IEEE Intelligent Systems**, v. 32, n. 2, p. 9–15, 2017. Citation on page [32](#).

MANDELBROT, B. B.; FREEMAN, W. H.; COMPANY. **The Fractal Geometry of Nature**. [S.l.: s.n.], 1983. (Einaudi paperbacks). Citation on page [34](#).

MEIER, A.; KRAMER, O. An experimental study of dimensionality reduction methods. In: **KI 2017: Advances in Artificial Intelligence**. [S.l.: s.n.], 2017. p. 178–192. Citation on page [48](#).

MOORE, G. E. Cramming more components onto integrated circuits. **Proceedings of the IEEE**, IEEE, v. 86, n. 1, p. 82–85, 1998. Citation on page [37](#).

NANDIMATH, J.; BANERJEE, E.; PATIL, A.; KAKADE, P.; VAIDYA, S.; CHATURVEDI, D. Big data analysis using apache hadoop. In: IEEE. **Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on**. [S.l.], 2013. p. 700–703. Citation on page [39](#).

NUNES, S. A.; ROMANI, L. A.; AVILA, A. M.; COLTRI, P. P.; TRAINA JR., C.; CORDEIRO, R. L.; SOUSA, E. P. de; TRAINA, A. J. Analysis of large scale climate data: How well climate change models and data from real sensor networks agree? In: **Proceedings of the 22Nd International Conference on World Wide Web Companion**. [S.l.: s.n.], 2013. (WWW '13 Companion), p. 517–526. Citation on page [36](#).

ORDOZGOITI, B.; CANAVAL, S. G.; MOZO, A. Massively parallel unsupervised feature selection on spark. In: **New Trends in Databases and Inf. Sys**. [S.l.: s.n.], 2015. p. 186–196. Citation on page [44](#).

PETSCHARNIG, S.; LUX, M.; CHATZICHRISTOFIS, S. Dimensionality reduction for image features using deep learning and autoencoders. In: **Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing**. [S.l.: s.n.], 2017. (CBMI '17), p. 23:1–23:6. Citation on page [36](#).

RADENSKI, A.; EHWERHEMUEPHA, L. Speeding-up codon analysis on the cloud with local mapreduce aggregation. **Information Sciences**, v. 263, n. 0, p. 175 – 185, 2014. Citation on page [39](#).

RAO, T. R.; MITRA, P.; BHATT, R.; GOSWAMI, A. The big data system, components, tools, and technologies: a survey. **Knowledge and Information Systems**, v. 60, n. 3, p. 1165–1245, 2019. Citation on page [37](#).

RATHORE, M. M.; AHMAD, A.; PAUL, A. Iot-based smart city development using big data analytical approach. In: IEEE. **Automatica (ICA-ACCA), IEEE International Conference on**. [S.l.], 2016. p. 1–8. Citation on page [41](#).

ROCHA, Á.; ADELI, H.; REIS, L.; COSTANZO, S. **Trends and Advances in Information Systems and Technologies**. [S.l.]: Springer International Publishing, 2018. (Advances in Intelligent Systems and Computing, v. 2). ISBN 9783319777122. Citation on page [29](#).

SAIDI, R.; NCIR, W. B.; ESSOUSSI, N. Feature selection using genetic algorithm for big data. In: HASSANIEN, A. E.; TOLBA, M. F.; ELHOSENY, M.; MOSTAFA, M. (Ed.). **The International Conference on Advanced Machine Learning Technologies and Applications (AMLT2018)**. Cham: Springer International Publishing, 2018. p. 352–361. ISBN 978-3-319-74690-6. Citation on page [37](#).

SAYED, S.; NASSEF, M.; BADR, A.; FARAG, I. A nested genetic algorithm for feature selection in high-dimensional cancer microarray datasets. **Expert Systems with Applications**, v. 121, p. 233 – 243, 2019. Citation on page [37](#).

SCHÖLKOPF, B.; SMOLA, A.; MÜLLER, K. R. Nonlinear component analysis as a kernel eigenvalue problem. **Neural Computation**, v. 10, n. 5, p. 1299–1319, 1998. Citation on page [45](#).

SCHROEDER, M. **Fractals, Chaos, Power Laws**. 6. ed. [S.l.]: W.H. Freeman, 1991. Citation on page [34](#).

SINGH, D.; REDDY, C. K. A survey on platforms for big data analytics. **Journal of Big Data**, Springer, v. 2, n. 1, p. 8, 2015. Citation on page [37](#).

SOUSA, E. P. M.; AL. et. A fast and effective method to find correlations among attributes in databases. **Data Min. and Knowl. Disc.**, v. 14, n. 3, p. 367–407, 2007. Citation on page [32](#).

Sun, Z.; Li, Z. Data intensive parallel feature selection method study. In: **International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2014. p. 2256–2262. Citation on page [25](#).

TIPPING, M. E.; BISHOP, C. M. Probabilistic principal component analysis. **Journal of the Royal Statistical Society: Series B (Statistical Methodology)**, Wiley Online Library, v. 61, n. 3, p. 611–622, 1999. Citation on page [44](#).

TRAINA JR, C.; TRAINA, A. J. M.; WU, L.; FALOUTSOS, C. Fast feature selection using fractal dimension. In: **SBDD**. [S.l.: s.n.], 2000. p. 158–171. Citations on pages [45](#) and [56](#).

_____. Fast feature selection using fractal dimension. **JIDM**, v. 1, n. 1, p. 3–16, 2010. Citations on pages [32](#), [34](#), [36](#), [45](#), and [56](#).

TUNG, A. K. H.; XU, X.; OOI, B. C. Curler: Finding and visualizing nonlinear correlation clusters. In: **SIGMOD**. [S.l.: s.n.], 2005. p. 467–478. Citations on pages [25](#) and [26](#).

WHITE, T. **Hadoop: The definitive guide**. [S.l.]: " O'Reilly Media, Inc.", 2012. Citation on page [39](#).

YEH, M. C.; LEE, I. H.; WU, G.; WU, Y.; CHANG, E. Y. Manifold learning, a promised land or work in progress? In: **IEEE ICME**. [S.l.: s.n.], 2005. Citation on page [48](#).

ZAHARIA, M.; CHOWDHURY, M.; DAS, T.; DAVE, A.; MA, J.; MCCAULEY, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: USENIX ASSOCIATION. **Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. [S.l.], 2012. p. 2–2. Citations on pages [40](#) and [41](#).

ZAKI, M.; MEIRA, W.; MEIRA, W. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. [S.l.]: Cambridge University Press, 2014. ISBN 9780521766333. Citation on page [30](#).

ZHANG, C.; NI, Z.; NI, L.; TANG, N. Feature selection method based on multi-fractal dimension and harmony search algorithm and its application. **International Journal of Systems Science**, v. 47, n. 14, p. 3476–3486, 2016. Citations on pages [31](#), [34](#), [36](#), and [45](#).

