


How Does Software Configuration Parameters Impact Job's Execution Time in Spark?

Maria C. L. Nunes   [Federal Uni. of San Francisco Valley | mcarolinalnunes@gmail.com]

Jairson B. Rodrigues  [Federal Uni. of San Francisco Valley | jairson.rodrigues@univasf.edu.br]

 Computer Engineering Collegiate, Federal University of the San Francisco Valley, Antônio Carlos Magalhães Avenue, 510, Country Club, Juazeiro, BA, 48.902-300, Brazil.

Received: 04 September 2024 • Accepted: 11 March 2025 • Published: 22 May 2025

Abstract Traditional centralized systems cannot deal with the big data context. Distributed computing platforms such as Apache Spark have been widely adopted, but configuring their parameters is challenging given the number of factors and their interactions. This work employs Design of Experiments (DoE) techniques to screening most relevant factors regarding execution time of a Naïve Bayes machine learning distributed task on a subset of the PT7 Web Corpus, which has 14.88 GB of data. Employing a fractional factorial design with 192 experimental units and linear regression techniques with backward elimination, we obtained (i) the most relevant factors based on statistical significance and (ii) a model capable of predicting execution time according to parameters' values in the analyzed context. Our results also include a visualization technique based on Inselberg's Parallel Coordinates to comprehend the impact on performance facing various configuration possibilities.

Keywords: Big Data, Design of Experiments, Apache Spark, Time Performance

1 Introduction

The first years of the 21st century were marked by a leap in the capacity to generate data on an unprecedented scale. Traditional tools, based on centralized computing systems, no longer addressed the storage, processing, and analysis requirements produced by the complexity of the big data context [Amato, 2017]. As a result, new tools began to take advantage of the computing capacity of clusters of distributed machines, organized to achieve horizontal scalability [Rodrigues, 2020]. Frameworks such as Apache Spark [Zaharia *et al.*, 2010] have been widely adopted by the market and academia in the application layer of such clusters as a computing engine for big data.

The performance of distributed tasks can be affected, among many reasons, by hardware configuration parameters related to each computing node, by the cluster architecture, by the volume of data or its nature, by the task to be executed and also by configuration parameters of the computing software (framework) itself. Spark has several configuration parameters related to the execution environment, memory management, and others [Chen *et al.*, 2016]. Certain configurations have an impact on the duration of the task, and the inappropriate combination of values can result in significant performance degradation. As an example, a different set of parameters configuration can optimize BigDataBench's Grep benchmark from 150s to less than 75s. [Wang *et al.*, 2016; Ahmed *et al.*, 2020]. Furthermore, using real-world case studies, Gounaris and Torres [2018] found an average improvement factor of 1.34 for music recommendation. Optimizing execution time becomes especially important when employing the cloud computing model, in which the monetary cost depends on the allocated resources and their pricing, often calculated per minute. Therefore, it is necessary to

identify the most relevant parameters to make the most of the specific context: cluster size, machine capacity, type of algorithm to be executed, volume and nature of the data [Nguyen *et al.*, 2018; Rodrigues *et al.*, 2021].

Testing the influence of each of the factors on the time variable would become costly or even unfeasible, since Spark has more than 180 configuration parameters [Wang *et al.*, 2016]. Taking only twelve parameters, for example, 4096 experimental units would be necessary to evaluate the influence of the main factors and their interactions up to the highest order. Furthermore, to minimize the noise caused by background variables in the response variable, it is recommended that experiments be carried out with at least three replications [Montgomery, 2017]. That is, the number of experimental units would increase even more.

That said, this research aimed to screen the software configuration factors that produce the greatest impact on the execution time of a machine learning classification task in Spark using the PT7 Web dataset [Rodrigues *et al.*, 2020]. To this end, methodologies based on DoE — Design of Experiments [Fisher, 1936] were used to collect experimental evidence in a controlled environment; carefully establishing the quantity and limits of the variables, as well as the appropriate organization, conduction, and collection of results. Finally, factors that do not produce a statistically significant impact on performance were excluded and a linear model was developed that estimates execution time.

It is important to emphasize that the scope of the research does not include performance analysis in terms of accuracy, ROC curve or other metrics related to the classification capacity of the algorithm itself. The effort was focused on performance analysis in terms of the execution time of the algorithm, running on the distributed cluster with the software configuration variations of the framework.

Our main contributions are:

- theoretical basis and practical demonstration of Design of Experiments to quantify the impact of software configuration factors on the execution time of distributed jobs in Spark;
- a visual tool based on parallel coordinates for fine-tuning configurations.

The rest of the paper is organized as follows: Section 2 lists the related works and strategies taken by other researchers. The concepts about big data, DoE, parallel coordinates, and linear regression are presented in Section 3. The methodology is described in Section 4. The results are described and discussed in Section 5. Finally, the conclusions are condensed in Section 6.

2 Related Works

The following papers have proposed different approaches to analyze the influence of hardware and software parameters on the performance of the Spark framework. Techniques include trial and error, simulation, machine learning, and Design of Experiments (DoE).

Based on trial-and-error methodology, Petridis *et al.* [2017] tested the performance of 12 (twelve) Spark parameters chosen from over 150 others based on the authors' knowledge and experience with the framework. The impact on execution time for each test was analyzed, and a sequential procedure for parameter tuning was obtained. The methodology was then improved to measure pairs of parameters [Gounaris and Torres, 2018]. Ahmed *et al.* [2020], also based on the trial-and-error method, analyzed 18 (eighteen) different parameters and showed that tuning for various types of applications and different data sizes can result in better performance.

Machine learning predictive models have been used to estimate which configuration among 13 (thirteen) Spark parameters would yield the best performance, improving task execution by up to 55% [Wang *et al.*, 2016]. Attribute selection algorithms have been used to estimate the execution time of Spark tasks for different parameters and values [Nguyen *et al.*, 2018]. Also, the Intel® CoFluent™ Studio tool was used to simulate a Spark cluster and measure the execution time by scaling 33 software parameters and five different hardware parameter groups [Chen *et al.*, 2016].

Moving on to DoE applications in Computer Science, research can be found in several fields such as an effort to identify the hyperparameters with the greatest impact on the execution of the Random Forest algorithm using the R language [Lujan-Moreno *et al.*, 2018].

At last, Rodrigues *et al.* [2021] applied DoE techniques to investigate the performance of Spark clusters using full and 2^k fractional factorial experimental designs. The authors screened data size and **hardware parameters**: number of nodes, and for each, the number of cores, the amount of RAM, and the number of SSD disks. The objective was to investigate how they affect execution time and monetary cost in a cluster running machine learning algorithms. The experiments highlighted a higher relevance with respect to

execution time related to the data size and the number of computing nodes. Two non-intuitive results were also found: the amount of RAM did not impact time or cost, while SSD disks were neutral concerning their contribution to time, only impacting the monetary cost.

Table 1 summarizes all the cited works with information about the tested parameters, the techniques used in the experiments, and their results. All improvements are described in comparison with the default configuration of the Spark parameters.

This paper differs from others by applying DoE to exclusively analyze **software parameters** of the Spark framework, obtaining a linear regression model with backward elimination. This approach aims to identify the most relevant parameters from a subset preselected by the authors and to assist in decision-making in adjusting the configuration for better performance in terms of time, maintaining fixed the hardware environment.

Design of Experiments was the methodological choice because it is effective to achieving the main goal of this research: identify the result of variation effects of multiple computational configuration scenarios, and to screening relevant factors on time performance, delivering safe statistical results at predictable intervals. DoE presents a generalizable method, applicable to any distributed algorithm in clusters for big data processing, with revealing results regarding the dynamics of effects between the configuration settings of platforms and their result on performance metrics.

3 Theoretical Framework

The research involved big data concepts, statistical techniques of experimental designs (DoE) and regression analysis; concepts that are discussed in the following subsections.

3.1 Big data and Apache Spark

The definition of big data is based on its three fundamental characteristics, known as the 3 V's of big data [Laney, 2001]. The first of these, volume, denotes data sets in large quantities. There is also the velocity at which these data are produced and must be analyzed before they become obsolete [Laney, 2001; Amato, 2017]. Finally, the many sources and diverse formats of the data characterize its variety.

Scientific projects, the increased use of social networks, the advent of the Internet of Things, the widespread use of multimedia communication — all these factors contribute to the big data phenomenon and data generation in such quantity and complexity that analysis becomes challenging [Hashem *et al.*, 2015; Simonet *et al.*, 2015]. Such characteristics pose issues when using traditional storage and processing techniques based on centralized computing. These challenges are sought to be solved through clusters of machines connected in a high-speed network capable of providing the necessary computing power and storage capacity [Amato, 2017]. This approach simplifies the resolution of scalability problems, promotes redundancy, and provides solutions with fault tolerance capacity [Rodrigues, 2020].

Table 1. Summary of related works, methodological approaches, and their results.

Authors	Parameters	Technique	Result
Petridis <i>et al.</i> [2017]	12 Spark parameters	trial-and-error	Execution time up to 10 times faster
Gounaris and Torres [2018]	12 Spark parameters	trial-and-error	At least 20% of performance improvement Up to 3% of performance improvement changing value of only one of analyzed parameters
Ahmed <i>et al.</i> [2020]	18 Spark parameters	trial-and-error	Average of 36% of performance improvement (with improvement directly proportional to data size)
Wang <i>et al.</i> [2016]	13 Spark parameters	machine learning	Identification of the most relevant parameters for each workload and performance improvement of up to 40%
Nguyen <i>et al.</i> [2018]	13 Spark parameters	machine learning / feature selection algorithm	Up to 71% performance improvement for PageRank benchmark
Chen <i>et al.</i> [2016]	33 software parameters and 5 different hardware parameter groups	Software simulation (Intel® CoFluent™ tool)	Identification of the most relevant parameters for the used dataset and optimization of algorithm with balanced accuracy improvement from 0,63 to 0,81
Lujan-Moreno <i>et al.</i> [2018]	7 Random Forest algorithm parameters	DoE / response surface methodology	Identification of relevant factors for investigated workloads and performance improvement of up to 70%
Rodrigues <i>et al.</i> [2021]	4 hardware parameters	DoE / factorial fractional design)	

Spark [Zaharia *et al.*, 2010] has emerged as a unified computing engine and a set of libraries for parallel data processing on clusters of computers. Typically, using the framework with its default configuration is sufficient to perform any analysis reliably. However, when more efficiency is needed, pipeline and shuffle operations, communication between drivers and workers, data distribution within the cluster, and many more, can have their behaviors adjusted by specific configuration parameters [Gounaris and Torres, 2018; Nguyen *et al.*, 2018].

3.2 Design of experiments (DoE)

Methodology introduced by Fisher [1936] to provide guidelines and tools for planning and conducting experiments, as well as for subsequent analysis of the results with scientific trustworthiness. They make it possible to analyze the influence of each factor under study and their combinations up to the highest order. They are capable of identifying the most relevant factors, quantifying the impact of the variation of their levels on the performance metric to be investigated, reducing the impact of background variables. It is also possible to reduce the number of experiments needed to test a hypothesis and, at the same time, ensure the reliability of the results at the cost of neglecting higher-order interactions. Finally, with the most relevant factors in hand, it is possible to perform a process to optimize the response of interest [Montgomery, 2017]. Below are some fundamental concepts for understanding the topic:

- **dependent variable** or **response** - variable of a test that one wishes to analyze;
- **independent variable** or **factor** - element of an experiment that can be modified in order to obtain a change

in the dependent variable;

- **experiment** - purposeful and controlled changes in the values of the independent variables of a system so that changes in the response can be identified;
- **experimental unit** - concrete entity used in the experiment on which some change is made;
- **level** - categorical, discrete or continuous values that a factor can assume;
- **main effect** - change in the response generated by the change in the level of a factor calculated for the levels of all other factors;
- **interaction** - occurs when the change in the dependent variable, when varying a level of factor, depends on the levels of other factors;
- **randomization** - random order of the executions of experimental units;
- **replication**: repeat, independently, the execution of each combination of factors and levels of an experiment.

Factorial designs are used in experiments in which there are several factors and it is necessary to analyze the impact of the main effects and their interactions on the response variable. To do this, in a series of experiments, all possible combinations of the levels of all factors are tested by measuring the response found in each situation [Montgomery, 2017]. The 2^k factorial design is a specific case of a factorial design. In this case, for k factors analyzed, their values are varied at two levels, with minimum and maximum values defined by the researcher. Thus, it is necessary to observe the result of $2 * 2 * 2 * \dots * 2 = 2^k$ experimental units multiplied by the number of replications (usually three). Generalizing the full 2^k factorial design, the canonical regression model can be represented by Equation 1:

$$y = \beta_0 + \sum_{j=1}^k \beta_j x_j + \beta_{12} x_{12} + \dots + \beta_{12\dots k} x_{12\dots k} + \epsilon \quad (1)$$

- y , measured response;
- β_0 , the intercept;
- $\beta_j, \beta_{12\dots k}$, the partial coefficients for the main effects and interactions;
- x_j , the factors that influence the response;
- $x_{12\dots k}$, the interaction of the factors analyzed and
- ϵ , the random error.

The number of effects for k factors can be calculated as follows: there are $\binom{k}{1}$ main effects, $\binom{k}{2}$ interactions between two factors, $\binom{k}{3}$ interactions between three factors, and so on. In general, we have $n_{eff} = C_{k,i} = \frac{k!}{i!(k-i)!}$, where n_{eff} is the number of possible effects on the response variable.

A full factorial design can be reduced to a 2^{k-p} fractional factorial design. A 2^{k-1} design requires half the runs of a full factorial experiment, a 2^{k-2} design requires one-fourth, a 2^{k-3} design requires one-eighth, and so on. However, there is the disadvantage of confounding factors, depending on the design **resolution**. Confounding occurs when the impact of main effects cannot be distinguished from higher-order effects. In this case, when there are confounding factors, it is not possible to identify their effects separately. The design **resolution** describes how these factors are associated:

- Resolution III: main effects are not confounded with each other, but can be confounded with second-order effects. Second-order effects, on the other hand, are confounded with each other;
- Resolution IV: main effects are not confounded with each other, nor with second-order effects. Second-order effects are still confounded with each other;
- Resolution V: main effects and second-order interactions are not confounded with any other main or second-order effect. However, second-order effects are confounded with third-order effects.

Usually, a fractional factorial experimental design with the highest possible resolution should be used — subject to limitations on the total execution time of the experiments and their cost [Rodrigues, 2020].

3.3 Selection of regressor variables

A linear regression model is used to describe the relationship between independent variables and thus enable the estimation of the output variable [Kutner, 2005]. Not all regressors, or variables, are necessary to compose the model that describes the output response. Furthermore, one of the objectives of this research is to highlight which factors have the greatest impact on the dependent variable. Therefore, it is important to select the regressor variables that will be incorporated into the model [Montgomery and Runger, 2003].

Among the different techniques that can be used, the backward elimination method was chosen for this research. This method is characterized by initially incorporating all independent variables and, step by step, removing one variable at a

time until the final regression model with the most important regressors is obtained [Montgomery and Runger, 2003].

3.4 Inselberg's parallel coordinates

Inselberg [1985] published his seminal paper describing the technique for representing multiple dimensions on a 2D plane mapping $R^n \rightarrow R^2$ by means of parallel coordinates:

(...) a coordinate system for Euclidean N -Dimensional space R^N is constructed. On the plane with xy -Cartesian coordinates, and starting on the y -axis, N copies of the real line, labeled x_1, x_2, \dots, x_N , are placed equidistant (one unit apart) and perpendicular to the x -axis. They are the axes of the parallel coordinate system and all have the same positive orientation as the y -axis. A point C with coordinates (c_1, c_2, \dots, c_N) is represented by the polygonal line whose N vertices are at $(i-1, c_i)$ on the x_i -axis for $i = 1, \dots, N$. In effect, a 1-1 correspondence between points in R^N and planar polygonal lines with vertices on x_1, x_2, \dots, x_N is established.

Figure 1 shows a one-to-one correspondence between points in R^N and planar polygonal lines with vertices on the parallel axes, where the polygonal line C represents the point $\bar{C} = (c_1, c_2, c_3, c_4, c_5)$, that is, in the example, it represents on 2D a R^5 space with five dimensions.

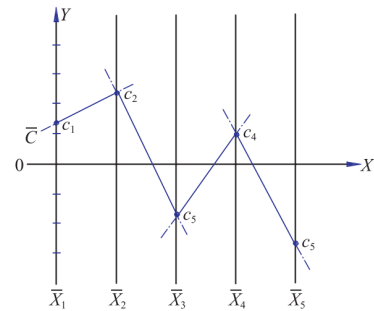


Figure 1. 2D representation of a R^5 space [Inselberg and Dimsdale, 2009].

Parallel coordinates applications are the most diverse and involve scientific visualization, including exploratory data analysis [Wegman, 1990] and visual multidimensional geometry [Inselberg and Dimsdale, 2009]. Regression is a common task for predicting the values of a dependent variable with respect to one or more independent variables. One can use parallel coordinates for visual regression [Wegman and Luo, 1997] or visualize statistical properties of regression models, such as exploratory modeling analysis [Unwin *et al.*, 2003], visual analysis approach for gene expression data [Dietzsch *et al.*, 2009], methods to visualize multivariate data about hurricane trends [Steed *et al.*, 2009], hardware performance for big data [Rodrigues *et al.*, 2021], etc. The seminal theory about parallel coordinates may be found at [Inselberg, 1985] and extensive surveys and applications may be found at [Inselberg and Dimsdale, 2009; Heinrich and Weiskopf, 2013].

4 Materials and Methods

This research employed clusters provisioned by the *Google Cloud Dataproc* service. Each of the 192 experimental units is related to the execution of the training and testing phases of a machine learning algorithm in an individual cluster, which was destroyed at the end of the collection of the metric of interest. Each cluster consisted of a master node and three worker nodes, equipped with 2 vCPUs and 8 GB of RAM. For storage, the master was provided with 100 GB and each worker with 200 GB. There was no hardware difference between the worker nodes. Network latency is a concern. However, when the nodes are connected to a high-speed internal network, it is expected that the network IO delays will be reduced. The DoE technique assumes that the experimental error of the model can be minimized by the randomness of the experimental plan.

We used Apache Spark v. 3.3.0 with Hadoop 3. The experiments were carried out by changing the Spark software configuration parameters when executing a Naïve Bayes multi-classification algorithm on the PT7 Web dataset¹ [Rodrigues *et al.*, 2020]. The original data was organized into 200 .parquet files. Ninety-five files were used, totaling 68.36 GB in their raw state, which, after preprocessing, resulted in 14.88 GB of input data for training and testing the machine learning model.

Table 2 details the configuration parameters chosen. They were chosen based on common configurations found in related works (Section 2) and in the official Spark configuration manuals. The minimum and maximum levels of the respective factors were determined after exploratory trials, observing the capacity limits of the cluster in question.

The classification task was performed considering seven classes, corresponding to the country of origin of the Portuguese text on each web page. To this end, the TLD (Top Level Domain) of the URL was used to separate texts from Brazil, Portugal, and other countries. The textual content of the pages was organized into sparse vectors derived from the processing of the original text, with irrelevant words (stop words) removed.

We selected the 2^k fractional randomized factorial design, $k = 7$ factors, resolution V, and three replications. In other words: a $2^{7-1}_{III} \times 3$ factorial design = 192 experimental units, as previously mentioned. The design was chosen due to the financial limit reserved for the project, while still ensuring results without confounding main factors with second-order interactions. With the execution of the experiments, the processing time of each job was collected. A linear regression model was adjusted with the results in hand. Thus, the influence of each factor was analyzed and, finally, the NIID premises² of the model's residuals were graphically verified.

¹Annotated corpus in Portuguese language composed of samples collected from seven countries: Angola, Brazil, Portugal, Cape Verde, Guinea-Bissau, Macau, and Mozambique.

² $\epsilon \approx NIID(0, \sigma^2)$: The variables of a linear model must be Normal, Independent and Identically Distributed (NIID) with mean equal to zero and variance σ^2 ; that is, unrelated random variables, approximated by a Normal distribution, exhibiting independence at all levels for the factors.

5 Results

Table 3 shows ten results³ regarding execution time in seconds for the experimental units of plan 2^{7-1}_{III} with three replications⁴. The first column shows the experimental unit number, while the second column shows the order in which each experimental unit was executed (also taking into account its replications). For example, experiment #1 had three replications, executed at the 53rd, 103rd and 159th positions. Next, there are the columns of the configuration parameters, and the column with the results obtained with the execution of the experiment, considering the value of the three replications. Finally, there is the column with the average results for each experiment.

5.1 Model adequacy checking and the Normality assumption

Table 4 shows the parameters present in the final model obtained by the backward elimination method⁵. The cutoff point for the significance level was 5% and all the variables in the model were statistically significant, as they presented $p\text{-value} < 0.05$.

Figures 2a and 2b show the histogram and the quantile-quantile plot of the residual distribution, respectively. The graphical analysis shows the Normal distribution format in the histogram. Furthermore, for the quantile-quantile plot, there are points far from the line considered outliers, but most of the points are gathered on the line of the theoretical quantiles, which shows Normality of the residuals.

Figure 3 shows the plots of Residuals *versus* Fitted Values (3a) and Studentized Residuals *versus* Fitted Values (3b). Here, it is possible to see the distribution of the residuals close to zero, with no notable patterns, evidence of homogeneity.

Finally, Figure 4 shows the residuals equally distributed. The plot with the regressors x_1x_7 shows a difference in distribution, but still acceptable for use in the model.

That said, based on the graphical analysis, one can assume the adequacy of the model to the basic assumptions of Normality, homogeneity of variance and independent distribution of its residuals.

5.2 Reduced Equation, Relevant Factors and Interactions

Table 4 shows the coefficients for each parameter estimate. Positive coefficients indicate an increase in the response, while negative coefficients indicate a decrease. The relevant factors, interactions, and their contributions are:

- positive impact of *spark.default.parallelism* (x_6);
- negative impact of interaction between *spark.shuffle.file.buffer* and *spark.default.parallelism* (x_1x_6);

³ The table with all the results has been summarized to save space.

⁴ $2^{7-1}_{III} \times 3$: randomized fractional factorial design, reduced by one-half with three replications.

⁵ Other techniques may be explored, such as information gain, random forests, and exhaustive selection.

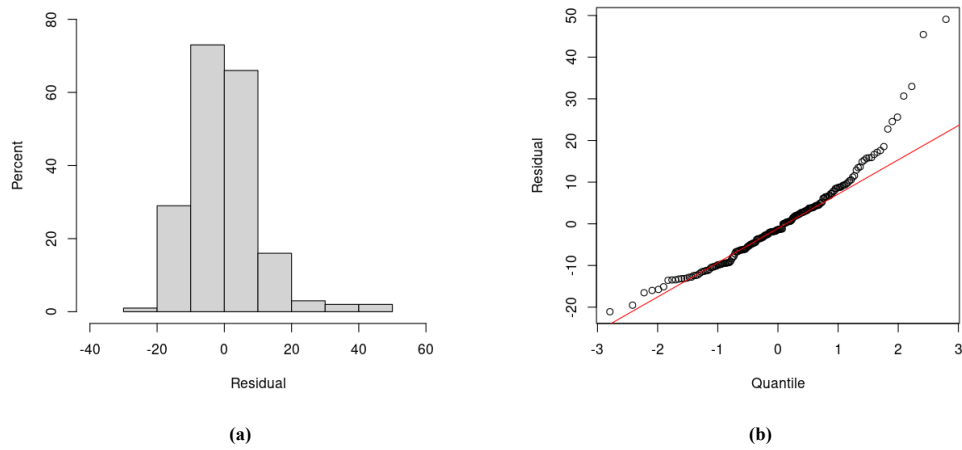


Figure 2. Normality of the residuals.

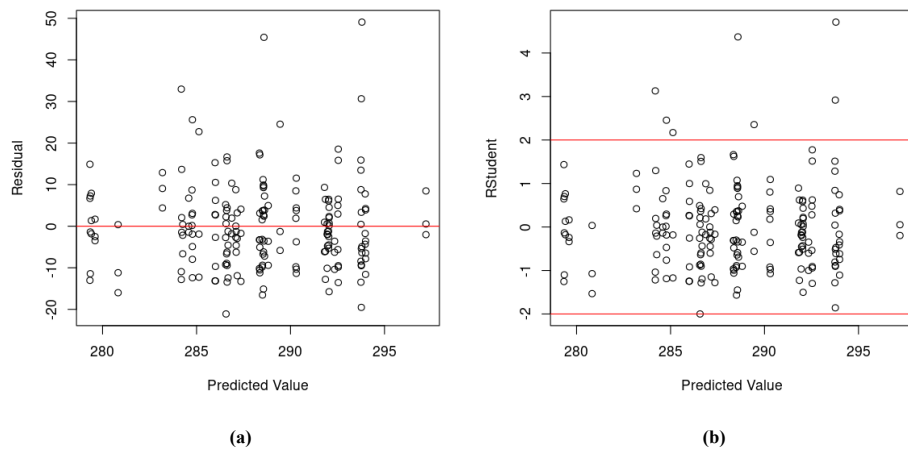


Figure 3. Plot of the residuals.

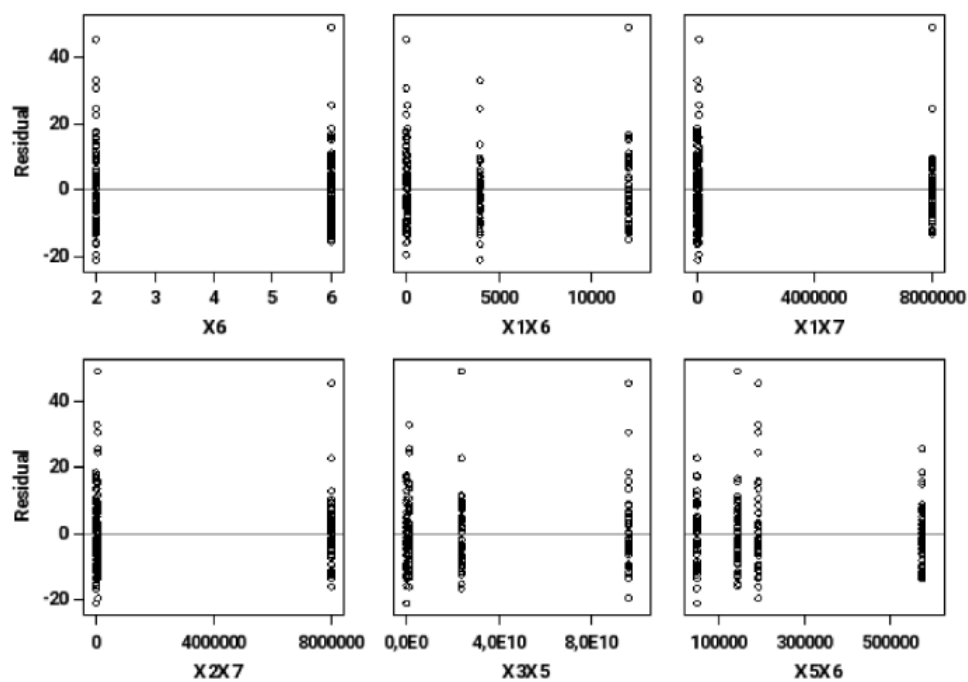


Figure 4. Residuals versus regressor variables.

Table 2. Spark software configuration parameters (factors).

Parameter	Levels	Symbol
<i>spark.shuffle.file.buffer</i>	16 KB and 2,000 KB	x_1
<i>spark.io.compression.lz4.blockSize</i>	16 KB and 2,000 KB	x_2
<i>spark.sql.files.maxPartitionBytes</i>	16,000 KB and 1,000,000 KB	x_3
<i>spark.sql.shuffle.partitions</i>	8 and 200	x_4
<i>spark.reducer.maxSizeInFlight</i>	24,000 KB and 96,000 KB	x_5
<i>spark.default.parallelism</i>	2 and 6	x_6
<i>spark.broadcast.blockSize</i>	16 KB and 4,000 KB	x_7

Table 3. Randomized $2^{7-1} \times 3$ fractional factorial design

#	Plan	x_1 (KB)	x_2 (KB)	x_3 (KB)	x_4	x_5 (KB)	x_6	x_7	t_1, t_2, t_3 (s)	\bar{t}
1	53-103-159	16 ⁻	16 ⁻	16000 ⁻	8 ⁻	24000 ⁺	2 ⁻	4000 ⁺	(277.16,291.50,283.29)	283.98
2	39-126-139	2000 ⁺	16 ⁻	16000 ⁻	8 ⁻	24000 ⁺	2 ⁻	16 ⁻	(265.46,288.82,287.18)	280.49
3	5-74-148	16 ⁻	2000 ⁺	16000 ⁻	8 ⁻	24000 ⁺	2 ⁻	16 ⁻	(278.18,305.93,285.07)	289.72
4	32-117-143	2000 ⁺	2000 ⁺	16000 ⁻	8 ⁻	24000 ⁺	2 ⁻	4000 ⁺	(273.18,291.06,277.19)	280.48
5	24-77-130	16 ⁻	16 ⁻	1000000 ⁺	8 ⁻	24000 ⁺	2 ⁻	16 ⁻	(279.98,294.08,294.70)	289.59
					[...]					
60	23-114-134	2000 ⁺	2000 ⁺	16000 ⁻	200 ⁺	96000 ⁺	6 ⁺	16 ⁻	(266.30,294.22,286.03)	282.18
61	7-87-160	16 ⁻	16 ⁻	1000000 ⁺	200 ⁺	96000 ⁺	6 ⁺	4000 ⁺	(278.98,308.38,283.05)	290.14
62	31-111-149	2000 ⁺	16 ⁻	1000000 ⁺	200 ⁺	96000 ⁺	6 ⁺	16 ⁻	(280.85,285.83,282.50)	283.06
63	16-76-167	16 ⁻	2000 ⁺	1000000 ⁺	200 ⁺	96000 ⁺	6 ⁺	16 ⁻	(297.51,295.45,286.90)	293.29
64	45-89-188	2000 ⁺	2000 ⁺	1000000 ⁺	200 ⁺	96000 ⁺	6 ⁺	4000 ⁺	(290.37,284.22,275.25)	283.28

Table 4. Parameter estimation.

Variable	Parameter estimate	Standard error	t-value	p-value
Intercept	286.49148	1.93574	148.00	<.0001
x_6	1.33830	0.53994	2.48	0.0141
x_1x_6	-0.00045818	0.00020747	-2.21	0.0284
x_1x_7	6.625831×10^7	2.835216×10^7	2.34	0.0205
x_2x_7	-6.52875×10^{-7}	2.427972×10^7	-2.69	0.0078
x_3x_5	8.21679×10^{-11}	2.17448×10^{-11}	3.78	0.0002
x_5x_6	-0.00001706	0.00000527	-3.24	0.0014

- positive impact of interaction between *spark.shuffle.file.buffer* and *spark.broadcast.blockSize* (x_1x_7);
- negative impact of interaction between *spark.io.compression.lz4.blockSize* and *spark.broadcast.blockSize* (x_2x_7);
- positive impact of interaction between *spark.sql.files.maxPartitionBytes* and *spark.reducer.maxSizeInFlight* (x_3x_5);
- negative impact of interaction between *spark.reducer.maxSizeInFlight* and *spark.default.parallelism* (x_5x_6).

It can be seen that the parameter *spark.sql.shuffle.partitions* (x_4) is not present in the model. In other words, considering all experimental variations and their interactions up to second order without confounding, the parameter proved to be irrelevant, not producing a statistically significant contribution, either positive or negative, against the value variations. Furthermore, taking into account the specific context (algorithm, data set and its volume), it can be noted that increasing the degree of parallelism leads to a worsening of performance, since an increase in the factor x_1 results in a positive contribution

to the response, that is, an increase in execution time. A practical interpretation for the positive impact of the first order factor *spark.default.parallelism* (x_6) resides in the definition of the configuration parameter itself in Spark, that is: “the default number of partitions in RDDs returned by transformations like *join*, *reduceByKey*, and *parallelize* when not set by the user”. In other words, the processing time increases as the number of partitions increases. Aside from x_6 , only second-order factors (interactions) were relevant. When interpreting results, the components of interactions have no practical interpretation [Montgomery, 2017, p. 411].

From the coefficients in Table 4 it is possible to construct a linear model to predict the execution time based on the relevant factors in the form of Equation (2):

$$t = 286.49 + 1.3383x_6 - 4.58E^{-4}x_1x_6 + 6.63E^{-7}x_1x_7 - 6.53E^{-7}x_2x_7 + 8.22E^{-11}x_3x_5 - 1.71E^{-5}x_5x_6 \quad (2)$$

The model showed statistical significance with $p\text{-value} = 0.0002 < 0.05$, $R^2 = 0.1320$ and adjusted $R^2_{adjusted} = 0.1038$. However, the low R^2 value indicates insufficient ex-

Table 5. Model variance analysis.

Variation source	DF	Sum of squares	Mean Square	F Value	p-value
Regression	6	3196.80166	532.80028	4.69	0.0002
Error	185	21028	113.6653		
Total	191	24225			

Table 6. Model analysis

RMSE	10.66140	R^2	0.1320
Mean of dependent variable	288.42790	$R^2_{adjusted}$	0.1038
Coefficient of variation	3.69638		

planatory capacity of the model. Other factors that may influence execution time are probably not included in the model. Although it does not invalidate the research findings, it suggests the need for in-depth exploration, new approaches with combinations of other factors, larger volumes of data, and new minimum and maximum limits for each factor. In addition, combined techniques such as the adaptive design of experiments based on Bayesian optimization [Rummukainen *et al.*, 2024], can offer greater flexibility and adaptability to complex environments with multiple interacting factors.

5.3 Visualization Tool to Aid Parameters' Configuration Process

Due to the wide range of parameters' values, it is difficult to understand the impact on performance of different parameter configuration scenarios. As described in Section 3.4, parallel coordinates can be useful for visualizing multidimensional data on a two-dimensional plane, especially when there are many instances in a problem. We use this technique for dynamic analysis of the impact on processing time caused by different values of software parameters between the lower and higher levels of the experimental design. Taking into account the levels defined in Table 2, we have the following possible configuration values ⁶ as input for the linear predictor model.

- x_1 , every 256 KB - 16, 256, 512, ... 2000;
- x_2 , every 512 KB - 16, 512, 1024, ... 2000;
- x_3 , every 150000 KB - 16000, 166000, ... 1000000;
- x_5 , every 12000 KB - 24000, 34000, ... 96000;
- x_6 , one by one - 2, 3, 4, 5, and 6;
- x_7 , every 512 KB - 16, 512, ... 4000.

All the above combinations produce $9 \times 5 \times 8 \times 7 \times 5 \times 9 = 113,400$ different configurations. Parallel coordinates can be used to isolate certain combinations and visually identify the impact on the dependent variable. Figure 5 shows that the time performance is between 285 and 290 seconds with respect to the configuration $[x_1, x_2, x_3, x_5, x_6, x_7] = [1536, 1024, 466000, \text{or } 616000, 36000 \text{ or } 48000 \text{ or } 60000, 5, 2048]$.

Similarly, Figure 6 shows how two different input values for *spark.shuffle.file.buffer* produce two different impacts on time performance: 284.09 and 286.67 seconds, respectively.

The configurations were $[x_1, x_2, x_3, x_5, x_6, x_7] = [512 \text{ or } 2000, 2000, 166000, 36000, 2, 4000]$.

Finally, the user may choose a specific time range and inspect all the possible configurations that produce that range. In Figure 7, the user asks the parallel coordinate system to exhibit all the parameter values that produce a processing time lower than 280 seconds. One can observe that it is impossible to reach the best performance while *spark.default.parallelism* < 5. Important: *spark.sql.files.maxPartitionBytes* and *spark.reducer.maxSizeInFlight* need to be set to maximum and minimum values, respectively.

These results are in accordance with the linear predictor model defined in Equation 2. All of these examples demonstrate the power of using parallel coordinates as a tool for visual linear regression and choosing configuration scenarios.

6 Conclusion

This work sought to identify relevant software configuration factors in the Spark distributed computing framework in the context of a document classification task with the Naïve Bayes algorithm on the PT7 WEB Corpus. Design of Experiments (DoE) techniques allowed valid conclusions to be drawn about the relevance of the factors analyzed in the research.

A linear regression model considering the main effects and second-order interactions was obtained from the results of the experiments. This model was statistically significant, with *p-value* < 0.05, showing how the execution time varies, indicating the most relevant factors and interactions. In this context, the research resulted in (i) the identification of the most relevant parameters (factors) for execution time, given the context; and (ii) a linear regression model that helps in understanding the variation in execution time based on the identified parameters. At last, we developed a visualization tool to help the user in the decision-making process face the many values (and its impact on response) between low and high levels of the tested experimental design.

The fact that the model was able to explain only 13% of the variation in the response raises evidence that other non-investigated factors are contributing to the execution time, pointing to directions for more comprehensive research. Important: when changing the dataset, the volume

⁶Intermediate values were arbitrarily defined by the researchers

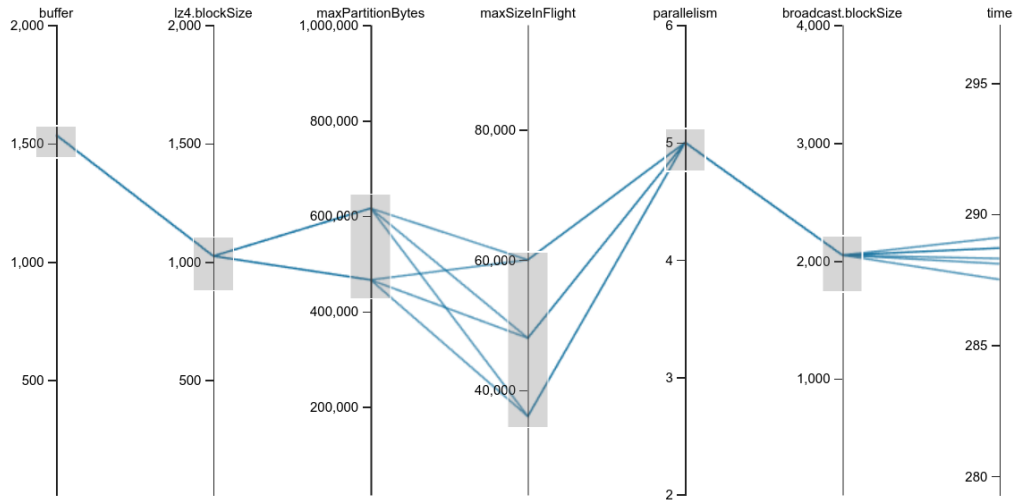


Figure 5. Time performance between 285 and 290 seconds regarding the configurations $[x_1, x_2, x_3, x_5, x_6, x_7] = [1536, 1024, 466000, \text{or } 616000, 36000 \text{ or } 48000 \text{ or } 60000, 5, 2048]$.

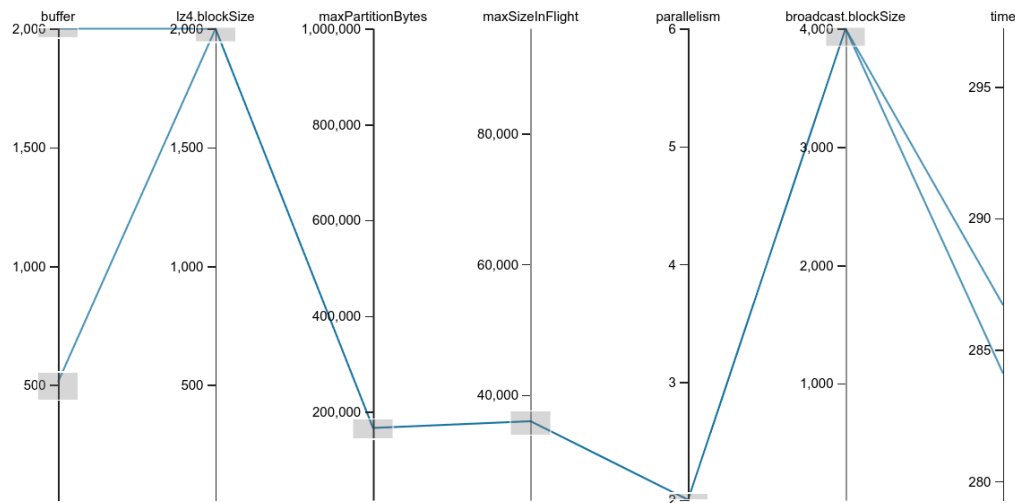


Figure 6. How two different input values for *spark.shuffle.file.buffer* impact time performance. Configurations: $[x_1, x_2, x_3, x_5, x_6, x_7] = [512 \text{ or } 2000, 2000, 166000, 36000, 2, 4000]$.

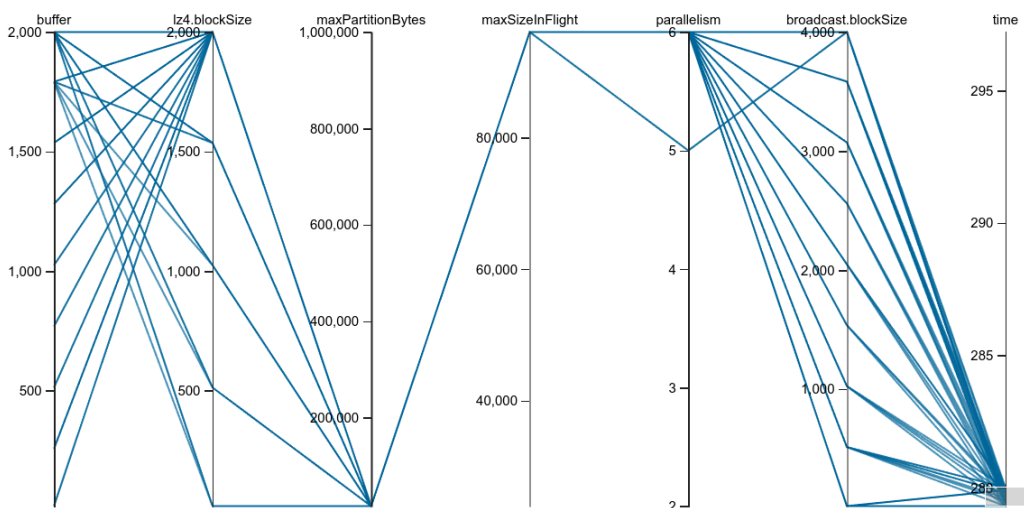


Figure 7. All configurations that result in execution time < 280 seconds.

of data, the nature of the task ⁷, or the experimental platform (especially hardware), the relevant parameters may also change. [Ahmed *et al.*, 2020].

Future work could investigate the **combined variation of software and hardware** parameters, including the number of nodes, the number of cores in each node, the amount of RAM, among others. It is also pertinent to analyze other dependent variables besides execution time, for example: disk usage rate, network traffic volume or CPU overhead. There are also more big data analysis tasks, such as: structured data, streaming, graph analysis or other machine learning algorithms, in clusters with greater computing power, with data volumes on scales greater than those analyzed in this paper.

Complementary techniques to the screening phase (2^k factorial designs) can be employed, such as Central Composite Design (CCD) and Box-Behnken. Such procedures extend the screening of relevant factors by adding axial and central points in the experimental plane in order to detect a non-linear relationship between the independent variables and the dependent variable and then apply the response surface methodology to optimize the output of interest [Lenth, 2009; Montgomery, 2017].

Finally, this paper points out paths for research and development of software libraries and applications centered on parallel coordinates as a tool to aid decision-making in the practical use of predictive linear models obtained through the Design of Experiments techniques.

Declarations

Acknowledgements

The authors are grateful for the access to laboratories, research materials, and technological resources provided by Federal University of San Francisco Valley throughout the course of this study.

Funding

The authors received no financial support for the research, authorship, and/or publication of this paper.

Authors' Contributions

JBR contributed to the conception of this study. MCLN performed the experiments. JBR and MCLN are the writers of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and materials

The data used in the ML tasks described in the paper are publicly available in the IEEE Data Port repository at: <https://ieee-dataport.org/open-access/pt7-web-annotated-portuguese-language-corpus>.

⁷A different machine learning algorithm or a task for other purposes.

References

- Ahmed, N., Barczak, A. L. C., Susnjak, T., and Rashid, M. A. (2020). A comprehensive performance analysis of apache hadoop and apache spark for large scale data sets using hibench. *J. Big Data*, 7(1):110. DOI: 10.1186/s40537-020-00388-5.
- Amato, A. (2017). *On the Role of Distributed Computing in Big Data Analytics*, pages 1–10. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-59834-5_1.
- Chen, Q., Wang, K., Bian, Z., Cremer, I., Xu, G., and Guo, Y. (2016). Simulating spark cluster for deployment planning, evaluation and optimization. In *2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, pages 1–11. Available at: <https://ieeexplore.ieee.org/document/7949406>.
- Dietzsch, J., Heinrich, J., Nieselt, K., and Bartz, D. (2009). Spray: A visual analytics approach for gene expression data. In *2009 IEEE Symposium on Visual Analytics Science and Technology*, pages 179–186. IEEE. DOI: 10.1109/VAST.2009.5333911.
- Fisher, R. A. (1936). *Design of experiments*, volume 1. BMJ Publishing Group. Book.
- Gounaris, A. and Torres, J. (2018). A methodology for spark parameter tuning. *Big Data Research*, 11:22–32. DOI: 10.1016/j.bdr.2017.05.001.
- Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., and Ullah Khan, S. (2015). The rise of “big data” on cloud computing: Review and open research issues. *Information Systems*, 47:98–115. DOI: 10.1016/j.is.2014.07.006.
- Heinrich, J. and Weiskopf, D. (2013). State of the art of parallel coordinates. *Eurographics (state of the art reports)*, pages 95–116. Available at: https://journals.de/files/heinrich_state_2013.pdf.
- Inselberg, A. (1985). The plane with parallel coordinates. *The visual computer*, 1:69–91. DOI: 10.1007/BF01898350.
- Inselberg, A. and Dimsdale, B. (2009). Parallel coordinates. *Human-Machine Interactive Systems*, pages 199–233. DOI: 10.1007/978-1-4684-5883-1.
- Kutner, M. (2005). *Applied Linear Statistical Models*. McGraw-Hill international edition. McGraw-Hill Irwin. Book.
- Laney, D. (2001). 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group. Available at: <https://diegonogare.net/wp-content/uploads/2020/08/3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>.
- Lenth, R. V. (2009). Response-surface methods in r, using rsm. *Journal of Statistical Software*, 32(7):1–17. DOI: 10.18637/jss.v032.i07.
- Lujan-Moreno, G. A., Howard, P. R., Rojas, O. G., and Montgomery, D. C. (2018). Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. *Expert Systems with Applications*, 109:195–205. DOI:

- 10.1016/j.eswa.2018.05.024.
- Montgomery, D. and Runger, G. (2003). *Estatística aplicada e probabilidade para engenheiros*. Livros Técnicos e Científicos. Book.
- Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley & sons. Book.
- Nguyen, N., Maifi Hasan Khan, M., and Wang, K. (2018). Towards automatic tuning of apache spark configuration. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 417–425. DOI: 10.1109/CLOUD.2018.00059.
- Petridis, P., Gounaris, A., and Torres, J. (2017). Spark parameter tuning via trial-and-error. In Angelov, P., Manolopoulos, Y., Iliadis, L., Roy, A., and Vellasco, M., editors, *Advances in Big Data*, pages 226–237, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-47898-2_4.
- Rodrigues, J., Vasconcelos, G., and Maciel, P. (2020). Pt7 web, an annotated portuguese language corpus.
- Rodrigues, J., Vasconcelos, G., and Maciel, P. (2021). Screening hardware and volume factors in distributed machine learning algorithms on spark: A design of experiments (doe) based approach. *Computing*, 103. DOI: 10.1007/s00607-021-00965-3.
- Rodrigues, J. B. (2020). *Análise de fatores relevantes no desempenho de plataformas para processamento de Big Data: uma abordagem baseada em projeto de experimentos*. Tese de doutorado, Universidade Federal de Pernambuco, Recife, PE, Brasil. Available at: <https://repositorio.ufpe.br/handle/123456789/39207>.
- Rummukainen, H., Hörhammer, H., Kuusela, P., Kilpi, J., Sirviö, J., and Mäkelä, M. (2024). Traditional or adaptive design of experiments? a pilot-scale comparison on wood delignification. *Heliyon*, 10(2). Available at: [https://www.cell.com/heliyon/fulltext/S2405-8440\(24\)00515-2?uuiid=uiid%3A35957045-7348-4ca0-897c-e65b3b392e1f](https://www.cell.com/heliyon/fulltext/S2405-8440(24)00515-2?uuiid=uiid%3A35957045-7348-4ca0-897c-e65b3b392e1f).
- Simonet, A., Fedak, G., and Ripeanu, M. (2015). Active data: A programming model to manage data life cycle across heterogeneous systems and infrastructures. *Future Generation Computer Systems*, 53:25–42. DOI: 10.1016/j.future.2015.05.015.
- Steed, C. A., Swan, J. E., Jankun-Kelly, T., and Fitzpatrick, P. J. (2009). Guided analysis of hurricane trends using statistical processes integrated with interactive parallel coordinates. In *2009 IEEE symposium on visual analytics science and technology*, pages 19–26. IEEE. DOI: 10.1109/VAST.2009.5332586.
- Unwin, A., Volinsky, C., and Winkler, S. (2003). Parallel coordinates for exploratory modelling analysis. *Computational Statistics & Data Analysis*, 43(4):553–564. DOI: 10.1016/S0167-9473(02)00292-X.
- Wang, G., Xu, J., and He, B. (2016). A novel method for tuning configuration parameters of spark based on machine learning. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 586–593. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0088.
- Wegman, E. J. (1990). Hyperdimensional data analysis using parallel coordinates. *Journal of the american statistical association*, 85(411):664–675. Available at: <https://www.tandfonline.com/doi/epdf/10.1080/01621459.1990.10474926?needAccess=true>.
- Wegman, E. J. and Luo, Q. (1997). High dimensional clustering using parallel coordinates and the grand tour. In *Classification and Knowledge Organization: Proceedings of the 20th Annual Conference of the Gesellschaft für Klassifikation eV, University of Freiburg, March 6–8, 1996*, pages 93–101. Springer. DOI: 10.1007/978-3-642-59051-1_10.
- Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*. Available at: https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/Zaharia.pdf.