Parâmetros de Configuração Relevantes para o Tempo de Execução de Tarefas no *Apache Spark*

Maria Carolina Lins Nunes[1], **Jairson Barbosa Rodrigues**[1]

[1]Universidade Federal do Vale do São Francisco (UNIVASF)
Av. Antônio Carlos Magalhães n. 510, Juazeiro, BA, Brasil – 48.902-300

mcarolinalnunes@gmail.com, jairson.rodrigues@univasf.edu.br

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 PT7 Web Corpus, having 14.88 GB of data. Employing a fractional factorial design with 192 experimental units and linear regression techniques with backward elimination, it was obtained a model capable of identif the most relevant factors regarging execution time in the analyzed context.*

Resumo. *Sistemas tradicionais centralizados não conseguem lidar com o contexto big data. Plataformas de computação distribuída como o Apache Spark têm sido amplamente adotadas, mas a configuração de seus parâmetros é desafiante face ao número de fatores e suas interações. Este trabalho emprega técnicas de Design of Experiments (DoE) para triar fatores de software mais relevantes para o tempo de execução de uma tarefa distribuída de aprendizagem de máquina Naïve Bayes sobre um subconjunto do Corpus PT7 WEB, com 14.88 GB de dados. Empregando um projeto fatorial fracionado com 192 unidades experimentais e técnicas de regressão linear com backward elimination obteve-se um modelo capaz de identificar os fatores mais relevantes para o tempo de execução de tarefas no contexto analisado.*

1. Introdução

Os primeiros anos do Século XXI foram marcados por um salto na capacidade de geração de dados em escalas sem precedentes. As ferramentas tradicionais, baseadas em sistemas computacionais centralizados, passaram a não mais endereçar as necessidades de armazenamento, processamento e análise geradas pela complexidade advinda do contexto *big data* [Amato 2017]. Com isso, surgiram ferramentas que passaram a tirar proveito da capacidade computacional de *clusters* de máquinas distribuídas organizadas de forma a obter escalabilidade horizontal [Rodrigues 2020]. *Frameworks* como o *Apache Spark* [Zaharia et al. 2010] têm sido amplamente adotados por mercado e academia na camada de aplicação de tais *clusters* como motor de computação sobre *big data*.

O desempenho de tarefas distribuídas pode ser afetado, dentre outras razões, por parâmetros de configuração de *hardware* relativos à cada nó de computação, pela arquitetura do *cluster*, pelo volume de dados ou sua natureza, pela tarefa a ser executada e

também por parâmetros de configuração do *software* (*framework*) de computação em si. O *Spark* possui diversos parâmetros de configuração relacionados ao ambiente de execução, gerenciamento de memória, dentre outros [Chen et al. 2016]. Certas configurações impactam o tempo de execução de uma tarefa e uma combinação de valores inadequada pode levar a uma perda de desempenho significativa [Wang et al. 2016]. A importância de otimizar o tempo de execução se torna evidente quando empregado o modelo de computação em nuvem, no qual o custo financeiro depende dos recursos provisionados e de sua tarifação, muitas vezes contabilizada por minuto. Por isso, é importante identificar os parâmetros mais relevantes para tirar o melhor proveito em relação ao contexto específico: tamanho do *cluster*, capacidade das máquinas, tipo de algoritmo a ser executado, volume e natureza dos dados [Rodrigues et al. 2021, Nguyen et al. 2018].

Testar a influência de cada um dos fatores sobre a variável tempo se tornaria custoso ou até inexequível, já que o *Spark* possui mais de 180 parâmetros de configuração [Wang et al. 2016]. Tomando-se apenas doze parâmetros, por exemplo, seriam necessários 4096 experimentos para avaliar a influência dos fatores principais e de suas interações até a mais alta ordem. Ainda, para minimizar a perturbação de variáveis de fundo (ruído) sobre a variável de resposta recomenda-se que experimentos sejam conduzidos com, no mínimo, três replicações [Montgomery 2017]. Ou seja, a quantidade de unidades experimentais aumentaria ainda mais.

Isto posto, esta pesquisa teve por objetivo triar os fatores de configuração de *software* que produzem maior impacto sobre o tempo de execução de uma tarefa de classificação de aprendizagem de máquina no *Spark* usando o conjunto de dados *big data PT7 Web* [Rodrigues et al. 2020]. Para tal, foram utilizadas metodologias baseadas em *DoE — Design of Experiments* [Fisher 1936] para coletar evidências experimentais em ambiente controlado; estabelecendo criteriosamente a quantidade e os limites das variáveis, bem como a apropriada organização, condução e coleta de resultados. Ao final, foram excluídos fatores que não produzem impacto estatisticamente significativo sobre o desempenho e desenvolvida uma função que estima tempo de execução na forma de uma equação linear reduzida composta pelos fatores relevantes.

Importa ressaltar que o escopo da pesquisa não inclui análises de desempenho em termos da acurácia, curva ROC ou outras métricas relativas à capacidade de classificação do algoritmo em si. O esforço foi concentrado na análise de desempenho em termos de tempo de execução do algoritmo em si, executando no *cluster* distribuído com as variações de configuração de *software* do *framework*.

Na sequência, o Capítulo 2 enumera os trabalhos relacionados, com exemplos de caminhos trilhados por outros pesquisadores. Os conceitos sobre *big data*, DoE e regressão linear são expostos no Capítulo 3 — Referencial Teórico. Já a metodologia empregada é descrita no Capítulo 4. Os resultados obtidos estão descritos e discutidos no Capítulo 5. E, por último, as considerações finais são condensadas no Capítulo 6.

2. Trabalhos relacionados

Os trabalhos aqui mencionados propuseram abordagens diversas para analisar a influência de parâmetros de *hardware* e de *software* sobre o desempenho do *framework Spark*. As técnicas variam da tentativa e erro, a simulações, passando por técnicas de aprendizagem de máquina e *Design of Experiments (DoE)*.

Uma metodologia baseada em tentativa e erro [Petridis et al. 2017] testou o desempenho de 12 (doze) parâmetros *Spark* escolhidos entre mais de 150 outros com base no conhecimento e experiência do autor com o *framework*. O impacto no tempo de execução para cada teste foi analisado e foi obtido um procedimento sequencial para ajuste de parâmetros. Em seguida, a metodologia foi aperfeiçoada para medir também pares de parâmetros [Gounaris and Torres 2018]. Outra pesquisa [Ahmed et al. 2020], usando também do método da tentativa e erro, analisou 18 parâmetros diferentes e mostrou que o ajuste para diferentes tipos de aplicação e diferentes tamanhos da massa de dados pode resultar em um melhor desempenho. Modelos preditivos de aprendizagem de máquina também já foram empregados para estimar com qual configuração dentre 13 (treze) parâmetros *Spark* se obteria o melhor desempenho, aperfeiçoando a execução da tarefa em até 55% [Wang et al. 2016]. Algoritmos de seleção de atributos foram utilizados para estimar o tempo de execução de tarefas *Spark* para diferentes parâmetros e valores [Nguyen et al. 2018]. De outra forma, a ferramenta *Intel® CoFluent™ Studio* foi utilizada para simular um *cluster Spark* e medir o tempo de execução com o dimensionamento de 33 parâmetros de *software* e cinco grupos de parâmetros de *hardware* diferentes [Chen et al. 2016].

Partindo para aplicações de DoE na Ciência da Computação, pode-se encontrar pesquisas em vários campos, como um esforço para identificar os hiperparâmetros de maior impacto sobre a execução do algoritmo Floresta Aleatória usando a linguagem R [Lujan-Moreno et al. 2018]. Foram conduzidos projetos experimentais fatoriais fracionados para triagem dos fatores mais importantes e depois um projeto fatorial completo para o ajuste de um modelo de segunda ordem usando técnicas de Metodologia de Superfície de Resposta [Tekindal et al. 2014].

Especificamente voltado para a investigação de parâmetros de *hardware* no *Spark*, utilizou-se DoE com projeto 2^k fatorial completo e fracionado para triar parâmetros de *hardware* na execução em *cluster* de algoritmos de aprendizagem de máquina sobre um conjunto de dados *big data* [Rodrigues et al. 2021].

Este trabalho difere dos demais ao aplicar DoE para analisar exclusivamente o impacto de parâmetros de *software* do *framework Spark*, obtendo um modelo de regressão linear com *backward elimination*. Essa abordagem tem o intuito de identificar os parâmetros mais relevantes de um subconjunto pré-selecionado pelos autores e auxiliar na tomada de decisão no ajuste das configurações de *software* do *framework* para melhor desempenho em termos de tempo, deixando as configurações de *hardware* fixas.

3. Referencial Teórico

A pesquisa envolveu conceitos de *big data*, técnicas estatísticas de projetos experimentais (DoE) e análise de regressão; conceitos discorridos nas subseções a seguir.

3.1. *Big Data* e *Apache Spark*

O conceito seminal de *big data* se baseia nas suas três características fundamentais, conhecidas como os 3 V's de *big data* [Laney 2001]. A primeira delas, o volume, denota conjuntos de dados em grandes quantidades. Há ainda a velocidade com que esses dados são produzidos e devem ser analisados antes de se tornarem obsoletos [Laney 2001, Amato 2017]. Por último, as muitas fontes de origem e diversos formatos dos dados caracterizam sua variedade.

Projetos científicos, a ampliação do uso de redes sociais, o advento da Internet das Coisas, o largo emprego de comunicação multimídia – todos esses fatores contribuem para o fenômeno *big data* e geram dados em quantidade e complexidade cuja análise torna-se desafiadora [Hashem et al. 2015, Simonet et al. 2015]. Tais características impõem desafios de manipulação quando empregadas técnicas tradicionais de armazenamento e processamento baseadas em computação centralizada. Busca-se solucionar tais desafios através de *clusters* de máquinas conectadas em uma rede de alta velocidade capazes de fornecer poder computacional e capacidade de armazenamento adequados [Amato 2017]. Esse tipo de abordagem facilita a resolução de problemas de escalabilidade, promove redundância e dota as soluções com a capacidade de tolerância a falhas [Rodrigues 2020].

O *Spark* [Zaharia et al. 2010] emergiu como um mecanismo de computação unificado e um conjunto de bibliotecas para processamento paralelo de dados em *clusters* de computadores. Normalmente, usar o *framework* com sua configuração padrão é o suficiente para fazer análises quaisquer de modo confiável. Porém, quando for necessária mais eficiência, operações de *pipeline* e *shuffle*, comunicação entre *driver* e *workers*, distribuição dos dados dentro do *cluster*, e muitas mais, podem ter seus comportamentos ajustados por parâmetros de configuração específicos [Gounaris and Torres 2018, Nguyen et al. 2018].

3.2. Projeto de Experimentos / *Design of Experiments (DoE)*

As técnicas de projetos experimentais (do termo em inglês *Design of Experiments (DoE)*) [Fisher 1936] fornecem diretrizes e ferramentas para o planejamento e condução de experimentos, bem como posterior análise dos resultados com segurança científica. Possibilitam analisar a influência de cada fator sob estudo e suas combinações até a mais alta ordem. São capazes de identificar os fatores mais relevantes, quantificando o impacto da variação de níveis destes sobre a métrica de desempenho que se deseja investigar, reduzindo o impacto de variáveis de fundo. É possível ainda, reduzir o número de experimentos necessários para se testar uma hipótese e, ao mesmo tempo, assegurar a confiabilidade dos resultados ao custo de negligenciar interações de mais alta ordem. Por fim, de posse dos fatores mais relevantes é possível realizar um processo de otimização da resposta de interesse [Montgomery 2017]. Na sequência, conceitos fundamentais para a compreensão do tema:

- **variável dependente** ou **resposta** - variável de um teste a qual se deseja analisar;
- **variável independente** ou **fatores** - elementos de um experimento que podem ser modificados a fim de se obter uma mudança na variável dependente;
- **experimento** - alterações propositais e controladas nos valores das variáveis independentes de um sistema para que se possa identificar as alterações na resposta;
- **unidade experimental** - entidade concreta utilizada no experimento sobre a qual alguma mudança é realizada;
- **nível** - valores categóricos, discretos ou contínuos que um fator pode assumir;
- **efeito principal** - mudança na resposta gerada pela mudança no nível de um fator calculado para os níveis de todos os outros fatores;
- **interação** - ocorre quando a mudança na variável dependente, ao variar um nível de um fator A, depende dos níveis de outros fatores;
- **randomização** - ordem aleatória das execuções individuais dos experimentos;

- **replicação**: repetir, independentemente, a execução de cada combinação de fatores e níveis de um experimento.

Projetos fatoriais são utilizados em experimentos nos quais existem vários fatores e é preciso analisar o impacto dos efeitos principais e suas interações sobre a variável de resposta. Para isso, em uma série de experimentos, todas as possíveis combinações dos níveis de todos os fatores são testados medindo a resposta encontrada em cada situação [Montgomery 2017]. O projeto fatorial 2^k é um caso específico de um projeto fatorial. Nesse caso, para *k* fatores analisados, seus valores são variados em dois níveis, com valores mínimo e máximo definidos pelo pesquisador. Assim, é necessário observar o resultado de $2 * 2 * 2 * ... * 2 = 2^k$ unidades experimentais multiplicada pelo número de replicações (normalmente três). Generalizando o projeto 2^k fatorial completo, o modelo de regressão canônico pode ser representado pela Equação 1 a seguir:

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

Sendo:

- y, resposta medida;
- β_0, o intercepto da reta;
- β_j, $\beta_{12...k}$, os coeficientes parciais para os efeitos principais e interações;
- x_j, os fatores que influenciam a resposta;
- $x_{12...k}$, a interação dos fatores analisados e
- ϵ, o erro aleatório.

O número de efeitos para *k* fatores pode ser calculado da seguinte forma: há $\binom{k}{1}$ efeitos principais, $\binom{k}{2}$ interações entre dois fatores, $\binom{k}{3}$ interações entre três fatores e assim sucessivamente. De forma geral, tem-se $n_{eff} = C_{k,i} = \frac{k!}{i!(k-i)!}$, onde n_{eff} é o número de efeitos possíveis sobre a variável de resposta.

Um projeto fatorial completo pode ser reduzido para um projeto 2^{k-p} fatorial fracionado. Um projeto 2^{k-1} precisa de metade das execuções de um experimento fatorial completo, 2^{k-2} precisa de um quarto, 2^{k-3} um oitavo, e assim por diante. Apesar disso, há a desvantagem do confundimento de fatores, dependendo da **resolução** do projeto. O confundimento ocorre quando não se pode diferenciar o impacto entre efeitos principais e outros de maior ordem. Nesse caso, quando há fatores confundidos, não é possível identificar o efeito destes separadamente. A **resolução** de um projeto descreve como esses fatores estão associados:

- Resolução III: efeitos principais não se confundem entre si, mas podem ser confundidos com efeitos de segunda ordem. Efeitos de segunda ordem, por sua vez, se confundem entre si;
- Resolução IV: efeitos principais não se confundem entre si, nem com efeitos de segunda ordem. Efeitos de segunda ordem ainda se confundem entre si;
- Resolução V: efeitos principais e as interações de segunda ordem não se confundem com nenhum outro efeito principal ou de segunda ordem. Porém, os efeitos de segunda ordem se confundem com os de terceira ordem;

Usualmente deve-se utilizar um projeto experimental fatorial fracionado com a maior resolução possível — ressalvadas as limitações de tempo total de execução dos experimentos e seu custo [Rodrigues 2020].

3.3. Seleção de Variáveis Regressoras

Um modelo de regressão linear serve para descrever a relação entre as variáveis independentes e assim, possibilitar a estimativa da variável de saída [Kutner 2005]. Nem todos os regressores, ou variáveis, são necessários para compor o modelo que descreve a resposta de saída. Ademais, um dos objetivos dessa pesquisa é destacar quais os fatores de maior impacto sobre a variável dependente. Com isso, é importante selecionar as variáveis regressoras que serão incorporadas ao modelo [Montgomery and Runger 2003].

Dentre diferentes técnicas que podem ser empregadas, escolheu-se para esta pesquisa o método *backward elimination*. Esse método se caracteriza por incorporar inicialmente todas as variáveis independentes e, passo a passo, retirar uma variável por vez até que seja obtido o modelo de regressão final com os regressores mais importantes [Montgomery and Runger 2003].

4. Materiais e Métodos

Esta pesquisa empregou *clusters* provisionados pelo serviço *Google Cloud Dataproc*. Cada uma das 192 unidades experimentais conduzida correspondeu à execução das fases de treinamento e teste de um algoritmo de aprendizagem de máquina em um *cluster* individual, sendo este destruído ao final da coleta da métrica de interesse. Cada *cluster* foi composto por um nó mestre e três nós trabalhadores, dotados de 2 vCPUs e 8 GB de RAM cada. Para o armazenamento, o mestre foi equipado com 100 GB e cada trabalhador com 200 GB. Foi utilizado o *Apache Spark* v 3.3.0 junto com o *Hadoop* 3. O experimento foi conduzido alterando os parâmetros de configuração de *software* do *Spark* na execução de um algoritmo de multiclassificação *Naïve Bayes* sobre o conjunto de dados PT7 Web [1] [Rodrigues et al. 2020]. Os dados originais estão dispostos em 200 arquivos no formato `.parquet`. Foram utilizados 95 destes somando 68.36 GB em estado bruto que, após pré-processamento, resultaram em 14.88 GB de dados de entrada para treinamento e teste do modelo de aprendizagem de máquina.

A tarefa de classificação foi realizada considerando sete classes, correspondentes ao país de origem do texto em Língua Portuguesa de cada página web. Para tal, foi utilizado o TLD (*Top Level Domain*) da URL para separar textos do Brasil, de Portugal e dos demais países. O conteúdo textual das páginas estava organizado em vetores esparsos derivados do processamento do texto original, removidas as palavras irrelevantes (*stopwords*).

O plano experimental escolhido foi o 2^k fatorial fracionado randomizado de resolução V, com k = 7 fatores, reduzido em uma fração e três replicações. Ou seja: um plano fatorial $2_V^{7-1} \times 3 = 192$ unidades expermientais, conforme mencionado anteriormente. A escolha do plano se deu pelo limite financeiro reservado ao projeto, garantindo ainda assim resultados sem confundimento dos fatores principais com interações de segunda ordem. A Tabela 1 detalha os fatores escolhidos (parâmetros de configuração de

[1] *Corpus* anotado em Língua Portuguesa composto por amostras coletadas de sete países: Angola, Brasil, Portugal, Cabo Verde, Guiné-Bissau, Macau e Moçambique.

Tabela 1. Parâmetros (fatores) de configuração de *software* do *Spark*

Parâmetro	Níveis	Símbolo
spark.shuffle.file.buffer	16KB e 2000KB	x_1
spark.io.compression.lz4.blockSize	16KB e 2000KB	x_2
spark.sql.files.maxPartitionBytes	16.000KB e 1.000.000KB	x_3
spark.sql.shuffle.partitions	8 e 200	x_4
spark.reducer.maxSizeInFlight	24.000KB e 96.000KB	x_5
spark.default.parallelism	2 e 6	x_6
spark.broadcast.blockSize	16KB e 4000KB	x_7

software do *Spark*). Tais fatores foram elencados com base em configurações comuns encontradas nos trabalhos relacionados (Capítulo 2) e nos manuais de configuração oficiais do *Spark*. Os níveis mínimo e máximo dos respectivos fatores foram arbitrados após execução exploratória dos autores, observados os limites da capacidade do *cluster* em questão. Com a execução dos experimentos seguindo o plano obtido, o tempo de execução de cada *job* foi coletado. De posse dos resultados, foi então ajustado um modelo de regressão linear. Com isso, a influência de cada fator foi analisado e, por fim, as premissas NIID [2] dos resíduos do modelo foram verificadas graficamente.

5. Resultados

Tabela 2. Projeto experimental $2_V^{7-1} \times 3$ para tempo de execução do *Naïve Bayes*

#	Plano	x_1 (KB)	x_2 (KB)	x_3 (KB)	x_4	x_5 (KB)	x_6	x_7 (KB)	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

A Tabela 2 mostra dez resultados[3] de tempo de execução em segundos para as unidades experimentais do plano 2_V^{7-1} com três replicações[4]. A primeira coluna mostra o número da unidade experimental, já a segunda coluna mostra a ordem na qual cada unidade experimental foi executada (levando em conta também suas replicações). Por exemplo, o experimento $\#1$ teve três replicações, executadas na 53ª, 103ª e 159ª posição do projeto. A seguir, tem-se as colunas dos parâmetros de configuração e, depois, a coluna com os resultados obtidos com a execução do experimento, considerando o valor das três replicações. Por último, tem-se a coluna da média dos resultados, para cada experimento.

[2] $\epsilon \approx NIID(0, \sigma^2)$: resíduos de um modelo linear devem ser **N**ormais, **I**ndependentes e **I**denticamente **D**istribuídos (**NIID**) com média zero e variância σ^2; ou seja, variáveis randômicas não relacionadas, aproximadas de uma distribuição Normal, exibindo independência em todos os níveis para os fatores.

[3] A tabela com todos os resultados foi suprimida por economia de espaço.

[4] 2_V^{7-1}: projeto fatorial completo 2^7 reduzido para um fracionado 2^6 com resolução V.

Tabela 3. Estimativas de parâmetro

Variável	Estimativa de parâmetro	Erro padrão	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

5.1. Análise de Pressupostos de Normalidade

A Tabela 3 mostra os parâmetros presentes no modelo final obtido pelo método *backward elimination*. O ponto de corte do nível de significância foi de 5% e todas as variáveis do modelo se mostraram estatisticamente significativas, pois apresentaram $p\text{-}value < 0,05$.

Tabela 4. Análise de variância do modelo

Fonte de variação	GL	Soma de quadrados	Quadrado médio	*F Value*	p-value
Regressão	6	3196,80166	532,80028	4,69	0,0002
Erro	185	21028	113,6653		
Total	191	24225			

Tabela 5. Análise do modelo

REQM	10,66140	R^2	0,1320
Média da variável dependente	288,42790	$R^2_{ajustado}$	0,1038
Coeficiente de variação	3,69638		

As Figuras 1a e 1b mostram o histograma e o gráfico quantil-quantil da distribuição dos resíduos, respectivamente. A análise gráfica evidencia o formato de distribuição Normal no histograma. Ainda, para o gráfico quantil-quantil, existem pontos afastados da reta considerados *outliers*, porém a maioria dos pontos está concentrado sobre a reta dos quantis teóricos, o que evidencia Normalidade dos resíduos.

A Figura 2 mostra os gráficos de Resíduos *versus* Valores Ajustados e o de Resíduos Estudentizados *versus* Valores Ajustados. Aqui, é possível ver a distribuição dos resíduos próximo do zero, sem padrões notórios, evidência da homogeneidade dos resíduos. Por último, a Figura 3, o gráfico de Resíduos versus Variáveis Regressoras mostra os resíduos igualmente distribuídos. O gráfico com os regressores x_1x_7 apresenta diferença na distribuição, porém ainda aceitável para uso do modelo.

Isto posto, com base na análise gráfica, pode-se assumir a adequação do modelo às suposições básicas de Normalidade, homogeneidade de variância e distribuição independente de seus resíduos.

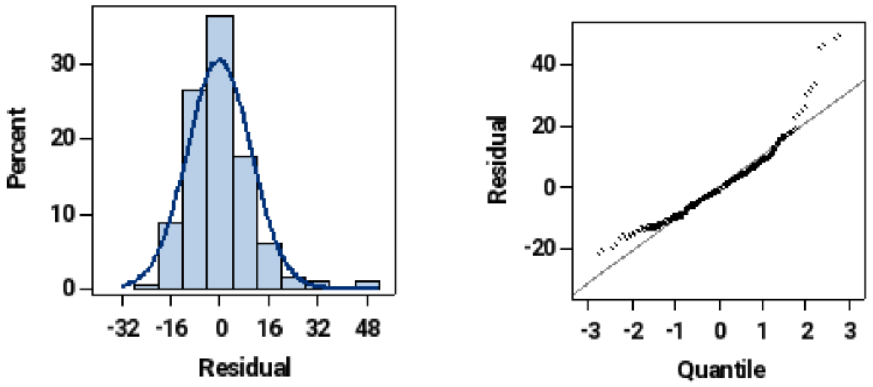

Figura 1. Normalidade dos resíduos

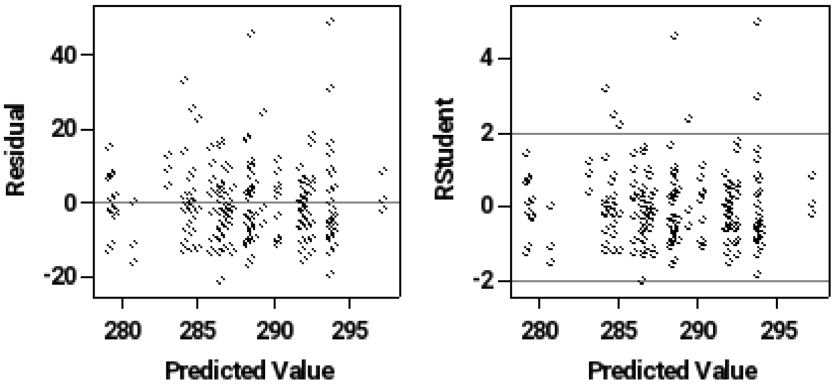

Figura 2. Gráficos de Resíduos *versus* Valores Ajustados

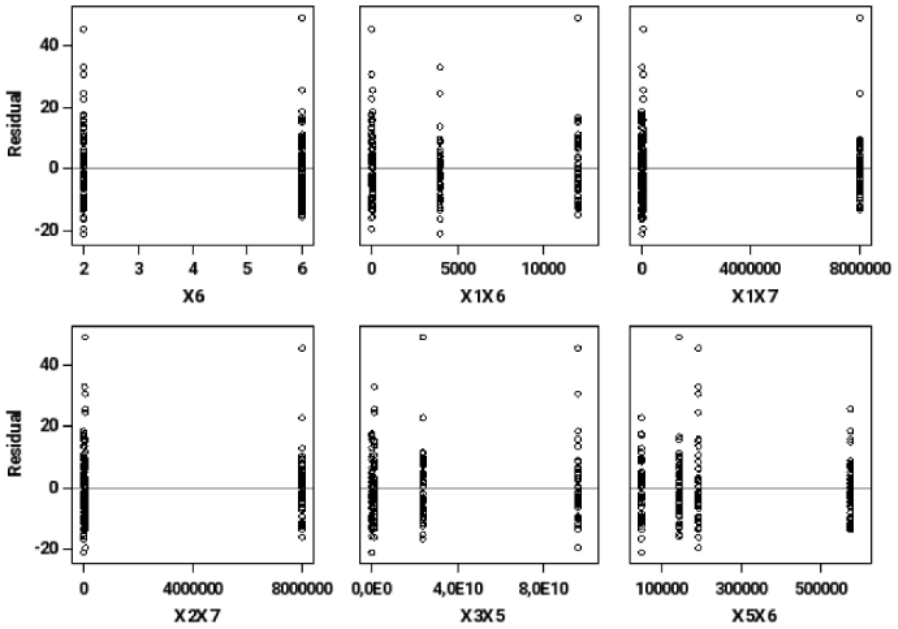

Figura 3. Gráficos de Resíduos *versus* Variáveis Regressoras

5.2. Equação Reduzia, Fatores e Interações Relevantes

Na coluna **Estimativa de parâmetro** da Tabela 3 é possível visualizar os coeficientes para cada variável. Coeficientes positivos indicam aumento no valor da variável dependente, enquanto coeficientes negativos indicam diminuição. Os fatores e interações relevantes e sua contribuição no tempo de execução são:

- impacto positivo do fator `spark.default.parallelism` (x_6);
- impacto negativo da interação `spark.shuffle.file.buffer` com `spark.default.parallelism` (x_1x_6);
- impacto positivo da interação `spark.shuffle.file.buffer` com `spark.broadcast.Ainda como oportunidade de estender o presente trabalhblockSize` (x_1x_7);
- impacto negativo da interação `spark.io.compression.lz4.blockSize` com `spark.broadcast.blockSize` (x_2x_7);
- impacto positivo da interação `spark.sql.files.maxPartitionBytes` com `spark.reducer.maxSizeInFlight` (x_3x_5);
- impacto negativo da interação `spark.reducer.maxSizeInFlight` com `spark.default.parallelism` (x_5x_6).

Percebe-se que o parâmetro `spark.sql.shuffle.partitions` (x_4) não está presente no modelo. Ou seja, considerando todas as variações experimentais e suas interações até segunda ordem sem confundimento, o parâmetro mostrou-se irrelevante, não produzindo contribuição estatisticamente significativa, seja positiva ou negativa, face às variações de nível. Ainda, levando em consideração o contexto específico (algoritmo, conjunto de dados e seu volume) nota-se que aumentar o grau de paralelismo leva a uma piora do desempenho, pois um aumento no fator x_1 resulta em contribuição positiva sobre a resposta, ou seja, aumento no tempo de execução.

A partir dos coeficientes da Tabela 3 é possível construir um modelo linear para previsão do tempo de execução com base nos fatores relevantes na forma da Equação 5.2:

$$t = 286,49148 + 1,33830x_1 - 0,00045818x_1x_6 + 6,625831 \times 10^{-6}x_1x_7 -$$

$$6{,}52875\times10^{-7}x_2x_7 + 8,21679 \times 10^{-11}x_3x_5 - 0,00001706x_5x_6 \quad (2)$$

O modelo mostrou significância estatística, com $p\text{-}value = 0,0002 < 0,05$. Todavia, apresentou valores aquém do satisfatório para explicar boa parte da variação sob a variável dependente com $R^2 = 0,1320$ e $R^2_{ajustado} = 0,1038$. Tal resultado indica a necessidade de investigações mais profundas, novas abordagens com combinações de outros fatores, maiores volumes de dados e novos limites mínimo e máximo para cada fator, sem com isso invalidar os achados da pesquisa.

6. Conclusão

Este trabalho buscou identificar fatores relevantes de configuração de *software* no *framework* de computação distribuída *Spark* no contexto de uma tarefa de classificação de documentos com o algoritmo *Naïve Bayes* sobre o *Corpus* PT7 WEB. Técnicas de *Design of Experiments (DoE)* possibilitaram tirar conclusões válidas sobre a relevância dos

fatores analisados na pesquisa. E um modelo de regressão linear considerando os efeitos principais e interações de segunda ordem foi obtido a partir dos resultados dos experimentos. Este modelo apresentou-se estatisticamente significativo, com $p\text{-}value < 0,05$, mostrando o modo como o tempo de execução varia, indicando os fatores e interações mais relevantes. Nesse contexto, a pesquisa teve como resultados (i) a identificação de parâmetros (fatores) mais relevantes para o tempo de execução, dado o contexto; e (ii) um modelo de regressão linear que auxilia no entendimento da variação do tempo de execução com base nos parâmetros identificados.

O fato do modelo ter conseguido explicar apenas 13% da variação sobre a resposta levanta evidências de que outros fatores não investigados estão contribuindo para o tempo de execução, apontando direções para pesquisas mais abrangentes. Deve-se ressaltar que ao mudar de aplicação, volume, natureza dos dados e plataforma experimental, principalmente *hardware*, os parâmetros de maior importância para variação do tempo de execução também podem mudar [Ahmed et al. 2020].

Trabalhos futuros poderão investigar a variação combinada de parâmetros de *software* e *hardware*, incluindo o número de nós, o número de núcleos em cada nó, a quantidade de memória RAM, dentre outros. Ainda é pertinente a análise de outras variáveis dependentes além do tempo de execução, por exemplo: taxa de uso de disco, volume de tráfego de rede ou sobrecarga de CPU.

Técnicas de DoE complementares à fase de triagem de fatores (projetos 2^k fatoriais) podem ser empregadas, a exemplo de *Central Composite Design — CCD* e *Box-Behnken*. Tais procedimentos estendem a triagem de fatores relevantes, adicionando pontos axiais e centrais no plano experimental a fim de detectar uma relação não linear entres as variáveis independentes e a variável dependente e então aplicar a metodologia da superfície de resposta para otimização da saída de interesse [Montgomery 2017].

Ainda há o emprego de outras tarefas de análise *big data*: SQL sobre *big data* estruturado, *Streaming*, análise de grafos ou outros algoritmos de aprendizagem de máquina, em *clusters* com maior poder de computação, com volumes de dados em escalas superiores às analisadas nesta pesquisa.

Referências

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.

Amato, A. (2017). *On the Role of Distributed Computing in Big Data Analytics*, pages 1–10. Springer International Publishing, Cham.

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.

Fisher, R. A. (1936). Design of experiments. *British Medical Journal*, 1(3923):554.

Gounaris, A. and Torres, J. (2018). A methodology for spark parameter tuning. *Big Data Research*, 11:22–32. Selected papers from the 2nd INNS Conference on Big Data: Big Data Neural Networks.

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.

Kutner, M. (2005). *Applied Linear Statistical Models*. McGrwa-Hill international edition. McGraw-Hill Irwin.

Laney, D. (2001). 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group.

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.

Montgomery, D. and Runger, G. (2003). *Estatística aplicada e probabilidade para engenheiros*. Livros Técnicos e Científicos.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

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.

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.

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.

Rodrigues, J. B. (2020). *Análise de Fatores Relevantes no Desempenho de Plataformas para Processamento de Big Data*. PhD thesis, Recife.

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.

Tekindal, M. A., Bayrak, H., Özkaya, B., and Yavuz, Y. (2014). Second-order response surface method: factorial experiments an alternative method in the field of agronomy. *Turkish Journal of Field Crops*, 19(1):35–45.

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.

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)*.