UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

MARIA CAROLINA LINS NUNES

Triagem de Fatores Relevantes para o Tempo de Execução de Processamento de Big Data: um Estudo de Caso com Parâmetros de Configuração no Apache Spark

JUAZEIRO - BA
2023

UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

MARIA CAROLINA LINS NUNES

Triagem de Fatores Relevantes para o Tempo de Execução de Processamento de Big Data: um Estudo de Caso com Parâmetros de Configuração no Apache Spark

Trabalho apresentado à Universidade Federal do Vale do São Francisco - Univasf, Campus Juazeiro, como requisito da obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Jairson Barbosa Rodrigues

JUAZEIRO - BA
2023

Nunes, Maria Carolina Lins
N972t Triagem de Fatores Relevantes para o Tempo de Execução de Processamento de Big Data: um Estudo de Caso com Parâmetros de Configuração no Apache Spark / Maria Carolina Lins Nunes. - Juazeiro - BA, 2023.
xii, 58 f.: il.; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Universidade Federal do Vale do São Francisco, Campus Juazeiro, 2023.

Orientador: Prof. Dr. Jairson Barbosa Rodrigues.

1.Processamento de dados. 2. Big Data. I. Título. II. Rodrigues, Jairson Barbosa. III. Universidade Federal do Vale do São Francisco.

CDD 004.6

Ficha catalográfica elaborada pelo Sistema Integrado de Biblioteca SIBI/UNIVASF
Bibliotecário: Márcio Pataro. CRB - 5 / 1369.

UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

FOLHA DE APROVAÇÃO

MARIA CAROLINA LINS NUNES

Triagem de Fatores Relevantes para o Tempo de Execução de Processamento de Big Data: um Estudo de Caso com Parâmetros de Configuração no Apache Spark

Trabalho apresentado à Universidade Federal do Vale do São Francisco - Univasf, Campus Juazeiro, como requisito da obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Jairson Barbosa Rodrigues

Aprovado em: 07 de março de 2023

Banca Examinadora

Documento assinado digitalmente
gov.br JAIRSON BARBOSA RODRIGUES
Data: 16/03/2023 19:41:36-0300
Verifique em https://validar.iti.gov.br

Jairson Barbosa Rodrigues, Doutor, Universidade Federal do Vale do São Francisco

Documento assinado digitalmente
gov.br RICARDO AZEVEDO MOREIRA DA SILVA
Data: 19/03/2023 19:36:56-0300
Verifique em https://validar.iti.gov.br

Ricardo Azevedo Moreira da Silva, Mestre, Universidade Federal do Vale do São Francisco

Documento assinado digitalmente
gov.br JORGE LUIS CAVALCANTI RAMOS
Data: 17/03/2023 10:37:09-0300
Verifique em https://validar.iti.gov.br

Jorge Luis Cavalcanti Ramos, Doutor, Universidade Federal do Vale do São Francisco

AGRADECIMENTOS

A Marilu, Péricles, Marilia e Guilherme, minha família, por sempre me apoiarem. Consigo sentir seu amor e zelo ao redor de mim, acompanhando tudo que faço.

A Bia, Alice e Henrique, meus sobrinhos, que me inspiram em ser sempre melhor.

Aos meus amigos, por me proporcionarem momentos de conforto e ensinamentos valiosos. A Ana Clara, que sempre me apoia incondicionalmente.

Aos meus colegas de curso, com os quais dividi todos esses anos, pelo suporte em momentos difíceis e regozijo em momentos felizes.

A todos os meus professores, pelo conhecimento enriquecedor compartilhado.

Ao meu orientador, Jairson, pelo suporte, paciência e ensinamentos que não se limitam às fronteiras da universidade.

Mas se o bastante de nós sonhar, se apenas mil de nós sonharmos, poderemos mudar o mundo.

Neil Gaiman
***Sandman* #18: *A dream of a thousand cats*, 1988**

RESUMO

Os primeiros anos do Século XXI foram marcados por um salto na capacidade de geração de dados em escalas sem precedentes na História. Em virtude disso, avanços combinados nas Tecnologias da Informação e Comunicação, na sociedade e na economia deram origem a um fenômeno comumente referenciado pelo termo *Big Data*. Nesse contexto, as ferramentas tradicionais, baseadas em sistema computacionais centralizados, passaram a não mais endereçar as necessidades de armazenamento, processamento e análise geradas pela complexidade oriunda de conjuntos de dados tão volumosos. Por consequência, surgiram, então, paradigmas e ferramentas que passaram a tirar proveito da capacidade computacional de poderosos *clusters* de máquinas distribuídas organizadas de forma a comportar aumento horizontal escalável. Dentre as várias ferramentas que surgiram, o Apache *Spark* tem sido bem adotado por indústria e academia. Porém, ainda que com ferramentas adequadas, a análise *Big Data* é uma tarefa que demanda tempo. Assim, os parâmetros de configuração do *Spark* afetam o tempo de execução, mas identificar as configurações mais adequadas, torna-se uma tarefa bastante difícil, em especial pela grande quantidade de fatores. Testar a influência de cada fator experimentalmente, e a interação entre eles, pode ser muito custoso. Nesse contexto, esse estudo teve como objetivo identificar os fatores mais relevantes para minimização do tempo de execução em *cluster* de máquinas, bem como obter modelo matemático para auxílio na tomada de decisão na configuração de plataformas para *Big Data*. Por isso, para análise do impacto dos parâmetros de configuração no tempo de execução de tarefas *Spark*, foi utilizada a técnica de delineamento fatorial fracionado e ajuste de modelo de regressão linear pelo método da eliminação passo atrás. Como resultado, foi obtido modelo de regressão linear significativo que selecionou os fatores mais importantes dentre os sete analisados.

Palavras-chave: *big data*, *spark*, delineamento experimental, delineamento fatorial fracionado.

ABSTRACT

The first years of the 21st century were marked by a leap in the ability to generate data on unprecedented scales in history. As a result, combined advances in Information and Communication Technologies, society and the economy gave rise to a phenomenon commonly referred to by the term Big Data. In this context, traditional tools, based on centralized computational systems, no longer address the storage, processing and analysis needs generated by the complexity arising from such voluminous datasets. Consequently, paradigms and tools emerged that began to take advantage of the computational capacity of powerful clusters of distributed machines organized in such a way as to support scalable horizontal increase. Among the many tools that have emerged, Apache Spark has been well adopted by industry and academia. However, even with adequate tools, *Big Data* analysis is a time-consuming task. Thus, Spark configuration parameters affect the runtime, but identifying the most suitable configurations becomes a very difficult task, especially due to the large number of factors. Testing the influence of each factor experimentally, and the interaction between them, can be very costly. In this context, this study aimed to identify the most relevant factors for minimizing the execution time in a cluster of machines, as well as to obtain a mathematical model to aid in decision making when configuring platforms for Big Data. For this reason, to analyze the impact of the configuration parameters on the execution time of Spark tasks, the fractional factorial design technique was used and the linear regression model was adjusted using the backward elimination method. As a result, a significant linear regression model was obtained that selected the most important factors among the seven analyzed.

Key-words: big data, spark, design of experiments, factorial fractional design.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HDFS	Hadoop Distributed File System (Sistema de Arquivo Distribuído Hadoop)
LHC	Large Hadron Collider (Grande Colisor de Hádrons)
LSST	Large Synoptic Survey Telescope (Grande Telescópio de Levantamento Sinóptico)
NB	Naïve Bayes
NIID	Normais, Independentes e Identicamente Distribuídos
RDD	Resilient Distributed Dataset (Conjunto de Dados Resilientes Distribuídos)
RSM	Response Surface Methodology (Metodologia de Superfície de Resposta)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
vCPU	Virtual Centralized Processing Unit (Unidade Central de Processamento Virtual)

SUMÁRIO

1 INTRODUÇÃO

Os primeiros anos do Século XXI foram marcados por um salto na capacidade de geração de dados em escalas sem precedentes na História. Avanços combinados nas Tecnologias da Informação e Comunicação, na sociedade e na economia deram origem a um grande volume de dados sendo gerados em alta velocidade (HASHEM et al., 2015). Porém, esse fenômeno não se limita só à geração de dados, a análise deles também se tornou importante para diversos setores como saúde, *business intelligence*, ciências, entre outros (JIN; KIM, 2018; DASH et al., 2019; KAMBATLA et al., 2014).

Dessa maneira, as ferramentas tradicionais, baseadas em sistema 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 então paradigmas e ferramentas que passaram a tirar proveito da capacidade computacional de poderosos *clusters* de máquinas distribuídas organizadas de forma a comportar aumento horizontal escalável (RODRIGUES, 2020).

A exemplo disso, cita-se o sistema de arquivos distribuído GFS (GHEMAWAT; GOBIOFF; LEUNG, 2003) e o modelo de programação *MapReduce* (DEAN; GHEMAWAT, 2008) que foram introduzidos pela empresa Google™ em resposta ao crescimento vertiginoso do volume de dados de seus produtos para abstrair detalhes de paralelização, balanceamento de carga e tolerância à falhas. Por conseguinte, Zaharia e outros (2012) apresentaram o conceito de *Resilient Distributed Datasets* (RDD's), uma abstração de memória distribuída que permite computação em memória tolerante à falhas. Surgiu, então, o *Spark*, um *framework* de código aberto para processamento paralelo muito utilizado para análise *Big Data*.

Posto isso, observa-se um contexto de avanços contínuos para atender às crescentes exigências de análise de dados da atualidade.

1.1 JUSTIFICATIVA

A respeito da otimização de recursos, o *framework Spark* possui diversos parâmetros de configuração relacionados ao ambiente de execução, gerenciamento de memória, entre outros (CHEN et al., 2016; SPARK, 2022b). Certas configurações impactam o tempo de execução de uma tarefa e uma configuração inadequada pode levar a uma perda de performance significativa, em termos de tempo (WANG; XU; HE, 2016).

Por exemplo, a importância em otimizar o tempo se torna evidente ao ser usado o modelo de computação em nuvem. Esse modelo se tornou uma alternativa viável para a análise *Big Data* justamente por suas características de flexibilidade, escalabilidade

e alta disponibilidade, compatíveis com a computação distribuída. Todavia, o custo financeiro de seu uso depende dos recursos provisionados e do tempo de execução das tarefas. Normalmente, algoritmos com tempo de execução alto se tornam mais custosos financeiramente frente àqueles com menor tempo de execução, fixados os parâmetros de *hardware*. Por exemplo, uma instância de 4 vCPUs e 16GB de RAM custa em torno de US$ 0,13 por hora. [1]

Por isso, é importante otimizar o conjunto de parâmetros 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; VASCONCELOS; MACIEL, 2021; NGUYEN; KHAN; WANG, 2018). Testar a influência no tempo de cada um dos fatores, se torna muito custoso, devido a sua quantidade, já que o *Spark* tem mais de 180 parâmetros de configuração (WANG; XU; HE, 2016). Tomando-se, por exemplo, doze parâmetros, seriam necessários 4096 experimentos para avaliar a influência de cada um de seus fatores principais e de suas interações até a mais alta ordem. Conforme Montgomery (2013), para minimizar a perturbação sobre a resposta devido a variáveis de fundo, recomenda-se, ainda, que experimentos estatísticos sejam conduzidos com, no mínimo, três replicações. Ou seja, a quantidade de unidades experimentais aumentaria ainda mais.

Com isso, esse projeto se justifica por buscar desenvolver modelos para a tomada de decisão sobre configuração de tarefas *Spark* no intuito de minimizar o tempo de execução de cargas distribuídas e, consequentemente, diminuir outros custos associados.

1.2 OBJETIVOS GERAIS

Identificar entre os fatores analisados, os mais importantes fatores de configuração de *software* do *framework Apache Spark*, sobre o tempo de execução de tarefas usando o conjunto de dados *Big Data* PT7 *Web* (RODRIGUES; VASCONCELOS; MACIEL, 2020) e gerar equação reduzida para a previsão da resposta de interesse (tempo de execução).

1.3 OBJETIVOS ESPECÍFICOS

- Coletar os tempos de execução para o experimento conduzido;
- Selecionar parâmetros de configuração mais relevantes;
- Obter modelo de regressão linear para estimar o tempo de execução de tarefas com base nos fatores mais relevantes e auxiliar na tomada de decisão.

[1] *AWS Pricing Calculator* e *Google Cloud Pricing Calculator*, acessado em maio de 2022.

1.4 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho se organiza da seguinte maneira: O Capítulo 2 contém referencial teórico, o qual apresenta a fundamentação teórica acerca das ferramentas e conceitos empregados. A seção de trabalhos relacionados está no Capítulo 3, ela apresenta exemplos de caminhos trilhados por outros pesquisadores. A seção de materiais e métodos é o Capítulo 4, nela se explica a metodologia aplicada na pesquisa. Os resultados obtidos estão descritos e discutidos no Capítulo 5. E, por fim, as considerações finais são feitas no Capítulo 6.

2 REFERENCIAL TEÓRICO

Nesse capítulo são apresentados conceitos e fundamentos teóricos relacionados a este trabalho. Primeiro, um estudo sobre computação distribuída e análise *Big Data* é apresentado. Em seguida, discorre-se sobre pontos importantes acerca do funcionamento e da arquitetura do *framework Spark*. Por fim, são detalhadas as bases estatísticas de delineamentos experimentais, método escolhido para a condução da pesquisa.

2.1 BIG DATA

Projetos científicos[2], a ampliação do uso de redes sociais, o advento da Internet das Coisas, o largo emprego de comunicação multimídia; todos esses fenômenos geram dados em quantidade e complexidade cuja análise torna-se desafiadora (HASHEM et al., 2015; SIMONET; FEDAK; RIPEANU, 2015). O conceito seminal de *Big Data* se insere justamente nesses cenários, através da caracterização baseadas nas suas três caracaterísticas fundamentais, conhecidas coloquialmente como os 3 V's de *Big Data* (LANEY, 2001).

A primeira delas o **volume**, denota conjuntos de dados em grandes quantidades e com tendência de aumentar ainda mais, alcançando a magnitude de Petabytes (PB). Há ainda a **velocidade** com que esses dados são produzidos, bem como a velocidade na qual os dados 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**. Com o passar do tempo, mais V's foram adicionados à definição de *Big Data* como a variabilidade, valor e veracidade (HASHEM et al., 2015)

Tais características impõem desafios de manipulação quando empregadas técnicas tradicionais de armazenamento e processamento baseadas em computação centralizada. Por isso, a computação distribuída busca 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). Assim, esse tipo de abordagem facilita a solução de problemas de escalabilidade, promove redundância e dota as soluções com a capacidade de tolerância à falhas (RODRIGUES, 2020).

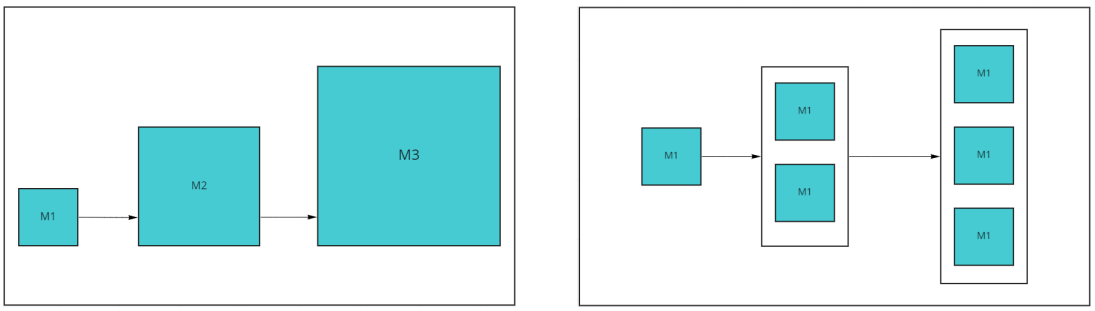
É importante esclarecer sobre os tipos de escalabilidade: há a escalabilidade vertical, a qual se refere à melhoria de performance de um sistema computacional por meio do aumento do poder computacional, da banda de conexão ou da memória e armazenamento desse sistema. Esse tipo de escalabilidade envolve a manutenção do número de instâncias do sistema, apenas modificando o poder computacional dessas instâncias como mostra a

[2] O LHC e o telescópio LSST são exemplos de instrumentos científicos que geram um grande volume de dados.

Figura 1a, onde M1, M2 e M3 representam sistemas computacionais diferentes com o poder computacional estando na ordem $M3 > M2 > M1$. Além disso, existe a escalabilidade horizontal, quando se adiciona mais recursos computacionais ao sistema em questão, nesse caso, aumentando o número de instâncias computacionais semelhantes, como mostra a Figura 1b (HUAI et al., 2011; SINGH; REDDY, 2014; TSAI et al., 2015).

Figura 1 – Tipos de escalabilidade

(a) Escalabilidade vertical **(b)** Escalabilidade horizontal



Fonte: A Autora (2023), adaptado de Tsai et al. (2015).

Dessa forma, com o tipo de arquitetura adequado, é possível extrair valor de grandes conjuntos de dados utilizando de *queries* SQL, algoritmos de aprendizagem de máquina, algoritmos de *MapReduce*, ferramentas para visualização desses dados por meio de grafos, entre outros. Para tanto, é necessário que as ferramentas usadas para executar essas tarefas façam uso adequado da arquitetura distribuída e consigam extrair dela todas as suas vantagens. Nesse contexto, é possível citar o paradigma *MapReduce*, e as ferramentas *Hadoop* e *Spark* como exemplos de tecnologias desenvolvidas e aprimoradas tendo como base a computação distribuída (AMATO, 2017).

2.2 *APACHE SPARK*

Trata-se de um mecanismo de computação unificado e um conjunto de bibliotecas para processamento paralelo de dados em *clusters* de computadores (SPARK, 2022b).

Conforme Salloum et al. (2015), ele se tornou uma ferramenta popular na indústria, sendo usado por empresas como IBM que criou o *SystemML*, biblioteca de código aberto para aprendizagem de máquina no *Spark*, Huawei que criou o Astro, um pacote que faz uso do SparkSQL, Yahoo, que usa essa ferramenta para aprendizagem profunda, entre outros.

O *Spark* foi apresentado para a comunidade por Zaharia et al. (2010) no artigo "*Spark: Cluster Computing with Working Sets*" (2010). Foi baseado no *Hadoop MapReduce* que, na época, era o mecanismo de programação paralela mais popular. Desde então, o projeto cresceu gradativamente com mais funcionalidades: ganhou a capacidade de programação interativa e *queries ad hoc*, suporte para *machine learning*, *streaming*, SQL e

análise de grafos, além de ganhar, também, interface de programação em Scala, Java e Python (CHAMBERS; ZAHARIA, 2018).

2.2.1 *Resilient Distributed Datasets* (RDD)

O conceito de RDD, introduzido por Zaharia et al. (2012), é uma coleção particionada de registros com permissão apenas para leitura. É possível obter um RDD a partir de dados em armazenamento ou de outro RDD, essas modificações em *datasets* são chamadas de transformações (ZAHARIA et al., 2012).

Essa abstração de dados é dita tolerante a falhas, isso se dá pelo conceito de linhagem, o qual se refere às informações sobre como foi obtido aquele *dataset* desde o conjunto de dados inicial. Com isso, o RDD é capaz de se reconstruir quando há ocorrências de falhas. Além disso, operações nesse tipo de *dataset* podem ser executadas em paralelo em todo o *cluster*, o que possibilita processamento rápido e escalável (SALLOUM et al., 2015; CHAMBERS; ZAHARIA, 2018).

Esse tipo de *dataset* é associado, principalmente, a três características. Há, primeiro, as dependências, essas informam ao *Spark* como reconstruir o *dataset* tendo como base o *dataset* pai, aqui reside o conceito de linhagem. Existem também as partições, que são partes atômicas do *dataset*, essa característica permite que o *Spark* paralelize a computação nos nós executores. Há também uma função de computação que produz um Iterator[T] para o dados que serão armazenados no RDD. Quando é necessário executar transformações e ações em um RDD, o planejador (*scheduler*) criará um grafo acíclico dirigido de estágios de execução (DAMJI et al., 2020; ZAHARIA et al., 2012).

2.2.2 Transformações e ações

No *Spark*, quando se deseja mudar uma estrutura de dados, um *dataset*, por exemplo, é necessário fazer uma transformação. Com uma transformação, um novo *dataset* com as mudanças desejadas é obtido, porém os dados originais não são alterados. Transformações são feitas de modo preguiçoso (do inglês, *lazy*), ou seja, não são executadas imediatamente: um plano de transformações é obtido e, só quando uma ação é chamada, esse plano é verdadeiramente executado. Esse plano faz parte da linhagem de um *dataset*. Essa abordagem permite que o *Spark* consiga otimizar essas transformações para uma execução mais eficiente, além de registrar o histórico de mudanças para possibilitar a tolerância à falhas (DAMJI et al., 2020; CHAMBERS; ZAHARIA, 2018; SALLOUM et al., 2015).

Transformações podem ser *narrow* (estreitas) ou *wide* (largas) como mostra a Figura 2. Na *narrow*, uma partição de saída pode ser obtida de apenas uma partição de entrada, um filtro é um exemplo de transformação *narrow*. Na *wide*, várias partições de entrada contribuem para várias partições de saída, exemplo desse tipo de transformação é a ordenação. Com isso, em transformações estreitas, o *Spark* executará uma operação de

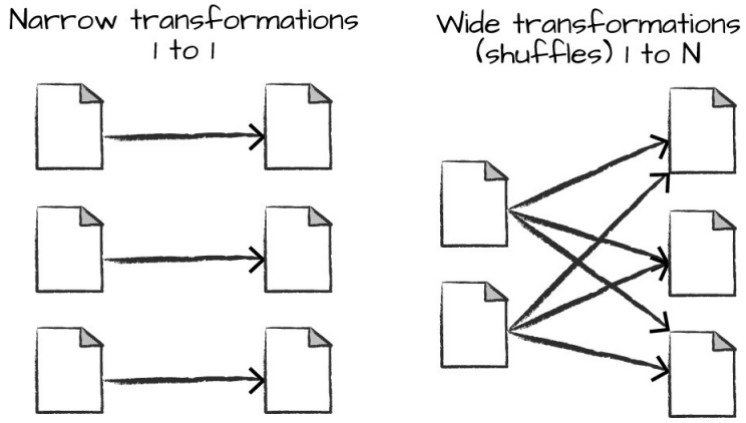
pipelining, isso significa que todas elas serão executadas em memória. Por outro lado, com transformações *wide*, o *Spark* executará a operação de *Shuffle*, nesse caso os resultados são escritos em disco. É esperado que a performance de transformações com *Shuffle* seja inferior às de operação com *pipeline*, devido às operações de entrada e saída em disco (DAMJI et al., 2020; CHAMBERS; ZAHARIA, 2018; SALLOUM et al., 2015).

As transformações com *Shuffle* podem ser configuradas de diversas maneiras, e uma configuração adequada pode diminuir o impacto do gargalo que as operações de entrada e saída em disco geram. Por exemplo, se pode citar alguns parâmetros de configuração que afetam operações de *Shuffle*, são eles: *spark.shuffle.file.buffer* e *spark.sql.shuffle.partitions* (DAMJI et al., 2020; CHAMBERS; ZAHARIA, 2018). Dessa forma, em uma aplicação com um grande número de transformações largas, se faz necessário uma maior atenção para esses parâmetros supracitados.

Figura 2 – Tipos de transformações do Spark

(a) Transformação do tipo *narrow*

(b) Transformação do tipo *wide*

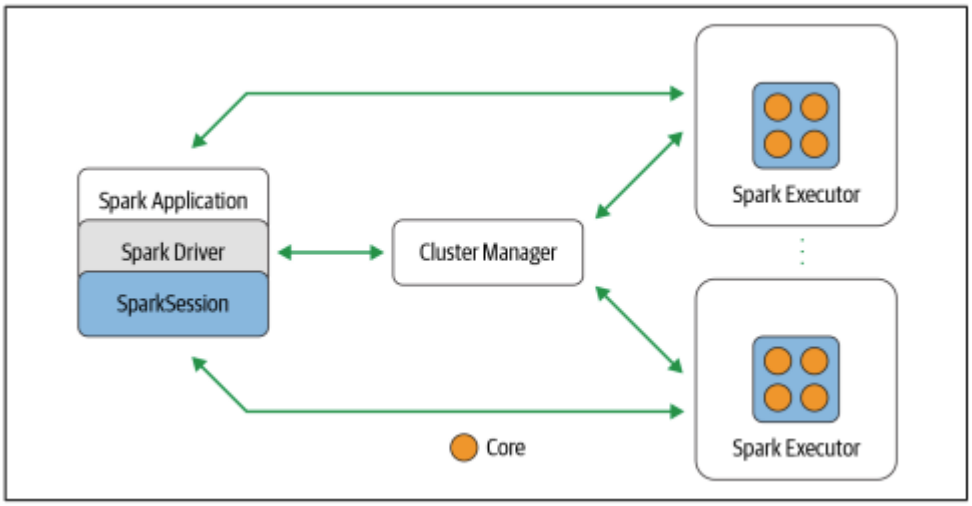


Fonte: Chambers e Zaharia (2018)

2.2.3 Arquitetura

O *Spark* utiliza uma arquitetura *master/worker*. Existe o processo *driver*, que é responsável pelo orquestramento das operações paralelas no *cluster*, e existe, também os processos *executors*, esses são responsáveis por executar, de fato, o trabalho o qual foram designados. Há também o gerente de *cluster* (*cluster manager*), responsável pelo gerenciamento e alocação de recursos para o *cluster* de nós em que ele está. O *Spark* tem seu *cluster manager* interno, porém também oferece suporte para o do *Apache Hadoop*, *YARN*, *Apache Mesos* e *Kubernetes* (DAMJI et al., 2020; CHAMBERS; ZAHARIA, 2018; SALLOUM et al., 2015).

Figura 3 – Arquitetura Spark



Fonte: Damji et al. (2020)

O processo *driver* executa a função *main*(). Ele fica em um nó do *cluster* e é responsável por manter informações sobre a aplicação, responder aos comandos do usuário e, por fim, analisar, distribuir e planejar o trabalho através do restante dos nós. Para tanto, ele se comunica com o *cluster manager* para alocação de recursos para os nós executores, transforma as operações em grafos acíclicos dirigidos (estágios de execução) e os distribui pelo *cluster*, comunicando-se com os nós executores para saber o estado da aplicação. Já a função de um processo *executor* é de executar o código designado a ele pelo processo driver. Além disso, ele reporta o estado da computação para o *driver* (DAMJI et al., 2020; CHAMBERS; ZAHARIA, 2018; SALLOUM et al., 2015).

2.2.4 Parâmetros de configuração

As operações de pipeline e de *Shuffle* citadas, bem como a comunicação entre *driver* e *workers*, a distribuição dos dados dentro do *cluster* e mais outras particularidades do funcionamento do *Spark* podem ter seus comportamentos modificados e ajustados, uma vez que, existem diversos parâmetros de configuração que mudam o funcionamento do *framework* e podem ser configurados para o objetivo que se queira alcançar.

Normalmente, usar a ferramenta com sua configuração padrão é o suficiente para fazer análises quaisquer de modo confiável, ou seja, é provável que um programa consiga ser executado até o final com parâmetros nos seus valores padrões (GOUNARIS; TORRES, 2018). Porém, pode ser necessário uma execução mais rápida ou mais eficiente. Nesse caso, é possível configurar a ferramenta de modo mais adequado para uma aplicação específica (GOUNARIS; TORRES, 2018; NGUYEN; KHAN; WANG, 2018).

É possível destacar algumas variáveis de configuração mais importantes. Por exemplo, as configurações destacadas por Damji et al. (2020) afetam, principalmente, o processo *driver*, os *executors* e o serviço de *Shuffle* que é executado em um nó executor. Além disso, autores como Petridis, Gounaris e Torres (2017), Ahmed et al. (2020), Wang, Xu e He (2016), entre outros, atestaram a influência de certos fatores no tempo de execução de aplicações Spark. Com base no trabalhos desses autores, se notabiliza a importância dos fatores descritos abaixo:

O *spark.shuffle.file.buffer* é o tamanho do buffer na memória para cada fluxo de saída de ums operação *Shuffle*. Isto é, dados intermediários são armazenados nesse *buffer*, e só depois escritos em disco, quando o *buffer* fica cheio. Um grande espaço alocado para esse parâmetro diminui a quantidade de escrita em disco, melhorando a performance dessas operações. Porém, essa alocação também pode tirar memória das outras tarefas e assim, prejudicar a performance (PETRIDIS; GOUNARIS; TORRES, 2017; DAMJI et al., 2020; CHEN et al., 2016).

Por sua vez, o *spark.io.compression.lz4.blockSize* dita o tamanho do bloco usado na compressão de dados feita pelo *codec* padrão do Spark. Nesse caso, aumentar esse número pode melhorar as operações de Shuffle, mas a compressão e descompressão de dados também compete por recursos (DAMJI et al., 2020; SPARK, 2022a).

O *Spark* processa dados em forma de partições no *cluster* de forma que cada tarefa agendada processa uma partição diferente, dessa forma cada partição pode ser vista como uma unidade atômica de paralelismo. O parâmetro que define o tamanho da partição de dados a ser processada é *spark.sql.files.maxPartitionBytes*. Um número alto para esse parâmetro pode comprometer o paralelismo do processo *Spark*, porém um número baixo pode acarretar em muitas operações de entrada e saída, prejudicando, também a performance (DAMJI et al., 2020).

As partições *Shuffle* são criadas nas transformações do tipo *wide*. Com isso, o resultado dessas operações são escritas nos discos dos executores. O número de partições criadas nesse processo é definida pelo parâmetro *spark.sql.shuffle.partitions*, esse número pode ser ajustado para reduzir o número de partições pequenas que são enviadas pela rede (DAMJI et al., 2020; ZAHARIA et al., 2010).

O parâmetro *spark.reducer.maxSizeInFlight* determina o tamanho máximo dos blocos de saída de operações *map* a serem coletadas simultaneamente de operações de *reduce*. Aumentar esse parâmetro acarreta em uma necessidade maior de memória, porque cada tarefa de *reduce* buscaria um bloco de dado maior, que por sua vez seria armazenado em *buffer* na memória (PETRIDIS; GOUNARIS; TORRES, 2017; GOUNARIS; TORRES, 2018; AHMED et al., 2020; WANG; XU; HE, 2016; NGUYEN; KHAN; WANG, 2018).

Já o *spark.default.parallelism* dita o número de partições de RDDs retornadas por

operações como *join* e *reduceByKey*. Similar ao *spark.sql.shuffle.partitions*, esse parâmetro também é relacionado ao paralelismo de operações de *Shuffle* e transformações *wide*, porém voltado a RDDs (DAMJI et al., 2020; AHMED et al., 2020).

Além disso, existem as variáveis de *broadcast* que são armazenadas em cache, mantidas em cada nó e que previnem que esses dados tenham que ser enviados como cópia para os executores junto da tarefa. O *Spark*, automaticamente, transmite dados comuns e reutilizáveis necessários para essas tarefas. Isto posto, o parâmetro de configuração *spark.broadcast.blockSize* define o tamanho do bloco de *broadcast*. Aumentar esse fator diminui o paralelismo, porém um número muito baixo pode prejudicar a performance do gerenciador de bloco (*BlockManager*) (SPARK, 2022a; NGUYEN; KHAN; WANG, 2018).

Os parâmetros supracitados foram escolhidos como parâmetros de interesse para execução dos experimentos desse trabalho e um resumo de suas descrições é mostrado na tabela 1.

Tabela 1 – Resumo dos parâmetros de configuração investigados

Parâmetro	Descrição
spark.shuffle.file.buffer	Tamanho do buffer na saída do *Shuffle*
spark.io.compression.lz4.blockSize	Tamanho do bloco de compressão
spark.sql.files.maxPartitionBytes	Tamanho das partições dos arquivos
spark.sql.shuffle.partitions	Número de partições *Shuffle*
spark.reducer.maxSizeInFlight	Tamanho máximo de saídas *map*
spark.default.parallelism	Partições retornadas (operações *Shuffle*)
spark.broadcast.blockSize	Bloco da variável de *broadcast*

Fonte: A Autora (2023).

No Capítulo 3, é mostrado que diversos autores conseguem melhorar a performance de tarefas *Spark* apenas mudando sua configuração. Pórem, apenas com os sete parâmetros de interesse escolhidos, seriam necessários $2^7 = 128$ experimentos, sem contar com replicações, para analisar o impacto da mudança de seus valores em dois níveis e todas as suas interações até a mais alta ordem, além disso a análise estatística dos dados coletados deve ser conduzida de forma que haja confiabilidade nos resultados (MONTGOMERY, 2013).

2.3 DELINEAMENTOS EXPERIMENTAIS

Delineamento experimental ou DoE (do inglês: *Design of Experiments*) se refere ao processo de planejamento de experimentos a fim de se coletar dados e analisá-los usando métodos estatísticos para se chegar a conclusões válidas e objetivas (MONTGOMERY, 2013).

As bases conceituais de DoE:

> "...foram construídas por Ronald Fisher e John Wishart (1930) em estudos nas áreas da Agricultura e Biologia, nos quais foram elaboradas ferramentas estatísticas para organizar experimentos capazes de lidar com a alta variação que torna confusa a compreensão de resultados observáveis e as condições que os provocam. Refinamentos da metodologia foram posteriormente desenvolvidos por estatísticos como George E. P. Box, Søren Bisgaard, William G. Hunter, e Genichi Taguchi; sendo atualmente uma das metodologias estatísticas mais utilizadas por engenheiros industriais na otimização do desempenho de processos por meio de configurações experimentais projetadas de maneira inteligente."
>
> (RODRIGUES, 2020, p. 59-60)

Dentre as diversas áreas de conhecimento nas quais esse método é aplicado podem-se citar a pesquisa farmacêutica (MISHRA et al., 2018; POLITIS et al., 2017); Engenharia (GUECIOUER; YOUCEF; TAREK, 2022); Agronomia (CASLER, 2015), Física (AMIN; KIANI, 2020), Inteligência Artificial (LUJAN-MORENO et al., 2018), dentre outras.

Uma estratégia para conduzir experimentos muito conhecida é a OFAT (*One Factor at a Time*). Consiste em variar um fator por vez enquanto os demais são mantidos constantes. A principal desvantagem reside no fato de que esse método não captura a interação entre fatores. Com a metodologia introduzida por Fisher (1935) e aprimorada por muitos outros autores, tais como Box e Meyer (1986) e Box e Wilson (1951), muitas variáveis são testadas de uma vez. Dessa forma é possível analisar a interação entre elas. Esta é a abordagem mais correta quando se lida com mais de um fator (MONTGOMERY, 2013; PACKIANATHER; DRAKE; ROWLANDS, 2000).

As técnicas de delineamento experimental (DoE) fornecem diretrizes e ferramentas para o planejamento experimental, condução dos experimentos e análise dos resultados com segurança científica. Possibilitam analisar a influência de cada fator em um projeto experimental 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. Fornecem ferramentas para reduzir e eliminar o impacto de variáveis de fundo (*nuisance factors*) no experimento. 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, 2013).

Alguns tipos de experimentos possíveis:

- Triagem de fatores: processo para entender quais fatores têm maior influência na resposta de interesse.
- Otimização: serve para identificar valores para os fatores que resultam na resposta desejada.

- Confirmação: processo para verificar que um sistema se comporta como diz alguma teoria ou experiência passada.
- Descoberta: nesse tipo de experimento, o experimentador que explorar novos materiais, novos fatores ou novos valores para os fatores.
- Robustez: serve para testar em que condições um sistema degrada.

Para cada um dos objetivos, há técnicas mais adequadas para serem empregadas (MYERS; MONTGOMERY; ANDERSON-COOK, 2016; RODRIGUES, 2020). Este trabalho tem como objetivo obter uma equação reduzida do tempo de execução em função dos parâmetros de configuração importantes, para que seja possível a previsão da resposta de interesse. Para isso serão utilizadas as técnica de delineamento 2^k fatorial fracionado e eliminação passo atrás. Esses conceitos são explicados nas subseções que se seguem.

2.3.1 Conceitos básicos

É importante esclarecer conceitos para compreensão das técnicas de delineamento experimental:

- **Efeito principal**: Ou efeito médio de um fator, como a média na mudança na resposta gerado pela mudança no nível de um fator calculado para os níveis de todos os outros fatores.
- **Experimento**: É um teste no qual são feitas alterações propositais e controladas nos valores das variáveis de um sistema para que se possa identificar as alterações que podem ser observadas na resposta de saída.
- **Interação**: Ocorre quando a mudança na variável dependente, ao variar um nível de um fator A, depende dos níveis de outros fatores.
- **Randomização**: é um conceito que faz com que a ordem das execuções individuais dos experimentos seja aleatória, isso faz com que a influência de fatores externos e variáveis de fundo seja minimizada.
- **Replicação**: significa repetir, independentemente, a execução de cada combinação de fatores e níveis. Esse princípio permite estimar o valor do erro experimental e permite, também, verificar que a diferença dos resultados observados é realmente diferente estatisticamente, ou seja, se não é apenas uma manisfestação do erro.
- **Variável dependente**: É o resultado ou a variável de saída de um teste a qual se deseja analisar.
- **Variável independente**: Elementos de um experimento que podem ser modificados a fim de se obter uma mudança na variável dependente.

2.3.2 Delineamento 2^k fatorial

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, 2013; MYERS; MONTGOMERY; ANDERSON-COOK, 2016).

O projeto fatorial 2^k é um caso específico de um projeto fatorial. Nesse caso, para os k fatores analisados, seus valores são variados em dois níveis. Com isso, para cada replicação do experimento, é necessário obter $2 * 2 * 2 * ... * 2 = 2^k$ observações.

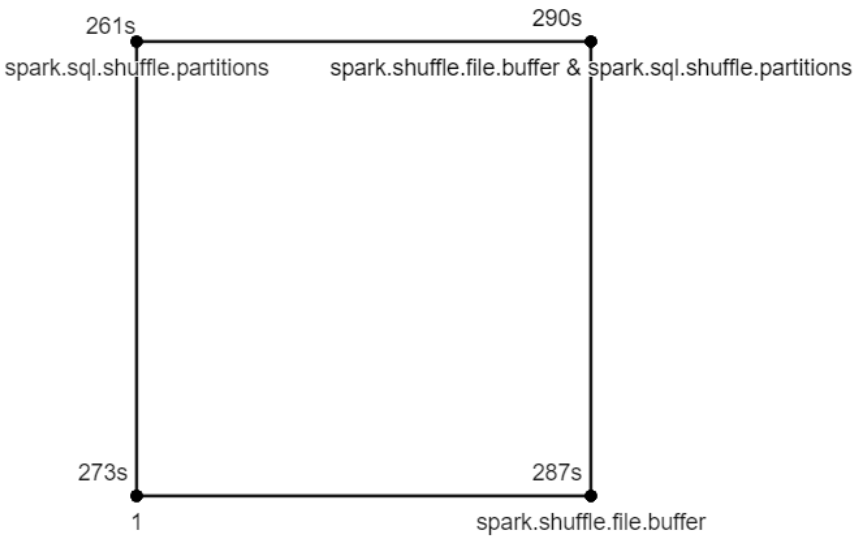
Para demonstrar os conceitos de **efeito principal** e **interação**, pode-se tomar uma situação hipotética onde existem os fatores *spark.shuffle.file.buffer* e *spark.sql.shuffle.partitions* e os níveis (-) e (+) para cada fator. A variável de resposta (tempo em segundos) para a variação dos níveis das variáveis independentes é mostrada na Figura 4. Para se representar o nível (+) do fator, utiliza-se seu nome e, quando os dois fatores estão em nível alto simultaneamente, seus nomes em conjunto. O símbolo (1) é usado para representar quando ambos os fatores estão em seu nível mais baixo. Para se calcular o efeito principal de um parâmetros, faz-se a média da diferença da resposta da variação do fator em questão, quando o outro está em nível alto somada à variação dos níveis de do fator em questão quando o outro está em nível baixo.

A demonstração do cálculo para o efeito principal de *spark.shuffle.file.buffer*, representado com a letra a pode ser vista na equação 2.1, já a equação 2.2 mostra o cálculo para o efeito principal *spark.sql.shuffle.partitions*, representado pela letra b. Ainda, há a equação 2.3 que mostra o cálculo da interação.

$$\begin{aligned} A &= \frac{1}{2n}\{[ab-b]+[a-(1)]\} \\ &= \frac{1}{2n}[ab+a-b-(1)] \\ &= \frac{1}{2\times 4}[290+287-261-273] \\ &= \frac{1}{2\times 4}\times 15 \qquad\qquad = 5,375 \end{aligned} \tag{2.1}$$

Da mesma forma, para o efeito principal de B:

Figura 4 – *Design* 2^2



Fonte: Autora (2023), adaptado de Myers, Montgomery e Anderson-Cook (2016).

$$\begin{aligned} B &= \frac{1}{2n}\{[ab-a]+[b-(1)]\} \\ &= \frac{1}{2n}[ab+b-a-(1)] \\ &= \frac{1}{2\times 4}[290+261-287-273] \\ &= \frac{1}{2\times 4}\times 19 \qquad\qquad = -1,125 \end{aligned} \tag{2.2}$$

A interação AB é a média das respostas dos níveis altos e baixos de AB simultaneamente menos a média das respostas dos níveis altos de A e B exclusivos. No gráfico, essa relação se demonstra no ponto direito superior do quadro e no ponto esquerdo inferior:

$$\begin{aligned} AB &= \frac{1}{2n}\{[ab+1)]-[a+b]\} \\ &= \frac{1}{2n}[ab+(1)-a-b] \\ &= \frac{1}{2\times 4}[290+273-261-287] \\ &= \frac{1}{2\times 4}\times 19 \qquad\qquad = 1,875 \end{aligned} \tag{2.3}$$

Nessa situação hipotética, é possível ver que o fator principal de *spark.shuffle.file.buffer* tem contribuição positiva para resposta, ou seja, quando ele aumenta, o tempo de execução também aumenta. Para o fator principal de *spark.sql.shuffle.partitions* é possível ver uma contribuição negativa, ou seja, quando ele aumenta, o tempo de execução diminui. A interação, por sua vez, tem contribuição positiva.

Outra maneira de se representar a variável dependente e seus fatores é por meio de um modelo de regressão. Nesse caso, esse modelo pode ser usado para estimar a resposta resultante para quaisquer valores das variáveis independentes. Tomando o exemplo acima e usando de dois fatores, o modelo de regressão linear com interações obtido a partir deles é mostrado na Equação 2.4

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \epsilon \tag{2.4}$$

Sendo:

- y, resposta medida;
- β_0, o intercepto da reta;
- β_1, β_2 e β_{12}, os coeficientes parciais para os efeitos principais e interações;
- x_1 e x_2, os fatores que influenciam a resposta;
- $x_1 x_2$, a interação entre os fatores;
- ϵ, o erro aleatório.

2.3.2.1 Delineamento 2^k fatorial genérico

É possível, então, generalizar o projeto de experimento 2^k fatorial completo. O modelo de regressão canônico pode ser representado pela Equação 2.5.

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \beta_{12} x_{12} + ... + \beta_{12...k} x_{12...k} + \epsilon \tag{2.5}$$

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 o ϵ, o erro aleatório.

O número de efeitos pode ser calculado da seguinte forma: para efeitos principais, faz-se $\binom{k}{1}$, para a interação entre dois fatores, faz-se $\binom{k}{2}$, entre três fatores, faz-se $\binom{k}{3}$ e assim sucessivamente. Generalizando, tem-se $n_{eff} = C_{k,i} = \frac{k!}{i!(k-i)!}$.

Como já dito, experimentos 2^k fatoriais são compostos de 2^k observações. Além disso, há, ainda, de se considerar replicações, que por sua vez são necessárias para estimar o erro aleatório e minimizar a interferência de varíaveis de fundo desconhecidas. Considerando as replicações, o número total de observações ao serem feitas pode ser calculada por $n_{obs} = n_r \times 2^k$. Com isso, é evidente que a se aumentar o número de fatores, o número de observações necessárias para um experimento fatorial se torna muito grande e, nem sempre, há recursos suficientes para tanto. Nesse caso, pode-se usar do **delineamento fatorial fracionado**, descrito na Seção 2.3.3 para diminuir o número de observações.

Com o delineamento 2^k fatorial é possível a obtenção de modelos que descrevem o sistema. Por exemplo, a Equação 2.5 é chamada de modelo de primeira ordem. Para usar esse modelo, assume-se que existe linearidade nos efeitos analisados. Porém, linearidade perfeita não é necessária e esse modelo ainda pode ser usado mesmo se a relação entre os efeitos e a resposta seja, apenas, aproximadamente linear. Além disso, a adição dos efeitos de interação faz o gráfico da superfície de resposta apresentar uma certa curvatura. Contudo, se a função de resposta não for aproximadamente linear, o modelo de primeira ordem não é adequado, nesse caso é necessário fazer uso do modelo de segunda ordem representado pela Equação 2.6.

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_{jj} x_j^2 + \epsilon \tag{2.6}$$

2.3.3 Delineamento 2^k fatorial fracionado

Com o aumento de fatores, o número de execuções necessárias para um experimento fatorial completo pode superar os recursos disponíveis do experimentador. Por exemplo, um experimento com seis fatores necessita de, no mínimo, $2^6 = 64$ execuções, sem contar com as replicações. Com o delineamento fatorial fracionado, é possível diminuir o número de observações necessárias ao descartar as interações de alta ordem da análise.

Apesar dessa negligência, essa técnica permite manter a confiabilidade dos resultados. Uma das ideias em que se baseia o uso desse método é o **princípio da esparsidade de efeitos**: quando há muitas variáveis, o desempenho medido do sistema provavelmente será influenciado em sua maioria pelos efeitos principais e interações de baixa ordem (MYERS; MONTGOMERY; ANDERSON-COOK, 2016).

Um projeto fatorial completo pode ser reduzido para um projeto 2^{k-p} fatorial fracionado. Um delineamento 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 sucessivamente.

Apesar disso, há a desvantagem do confundimento de fatores, a depender da **resolução** do delineamento.

O confundimento de fatores acontece quando não se pode diferenciar entre efeitos principais e os de ordem maior. Nesse caso, quando há fatores confundidos, não é possível identificar o efeito destes separadamente. A **resolução** de um delineamento descreve como esse fatores estão associados:

- Resolução III: nessa resolução, os efeitos principais não se confundem entre si, mas podem ser confundidos com efeitos de segunda ordem. Os efeitos de segunda ordem, por sua vez, se confundem entre si;
- Resolução IV: os efeitos principais não se confundem entre si, nem com os efeitos de segunda ordem. Os efeitos de segunda ordem ainda se confundem entre si;
- Resolução V: Os 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;

Recomenda-se usar da maior resolução possível para planejar os experimentos fracionados. Dessa forma, as suposições acerca de qual ordem de interações pode ser negligenciadas se tornam menos restritivas.

2.4 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, além disso, um dos objetivos dessa pesquisa é destacar quais os fatores de maior importância na variável dependente. Com isso, é importante selecionar as variáveis regressoras que serão incorporadas ao modelo (CHARNET REINALDO; FREIRE, 1999; MONTGOMERY; RUNGER, 2003).

Nesse contexto, há técnicas diferentes que podem ser empregadas. A técnica escolhida da pesquisa é o método da eliminação passo atrás (*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. Para tanto, a cada etapa o modelo completo é comparado com o modelo reduzido, este obtido pela retirada de uma determinada variável (CHARNET REINALDO; FREIRE, 1999; MONTGOMERY; RUNGER, 2003). Para fazer essa comparação, é observada a soma de quadrados de regressão extra e a estatística do teste de sua contribuição, dada pela Equação 2.7

$$\frac{SQR^c_{eg} - SQR^r_{eg}}{\widehat{\sigma^2}} \tag{2.7}$$

Sendo SQR^c_{eg} a soma de quadrados de regressão do modelo completo, SQR^r_{eg} a soma de quadrados de regressão do modelo reduzido e $\widehat{\sigma^2}$ o estimador de σ^2 correspondente ao ajuste do modelo completo. Cada etapa do procedimento tem os passos abaixo:

1. Ajustar modelo com todas as variáveis (modelo completo) e obter SQR^c_{eg} e $\widehat{\sigma^2}$
2. Para cada variável presente, ajustar modelo com a retirada dessa variável e calcular SQR^r_{eg} e a estatística com base na Equação 2.7
3. Identificar quais dos modelos reduzidos tem o valor mínimo (F_{min}) calculado no passo 2.
4. Comparar este valor com o quantil especificado da distribuição F (F_out). Se $F_{min} < F_{out}$, retira-se a variável correspondente ao valor de F_{min} e o processo todo se repete. Caso contrário, interrompe-se o processo e opta-se pelo modelo considerado nesta etapa.

Após obtenção do modelo final, é necessária a análise de resíduos que é essencial na avaliação do ajuste de modelo de regressão linear, além de contribuir para avaliar a importância das variáveis regressoras (CHARNET REINALDO; FREIRE, 1999).

2.4.1 Análise de resíduos

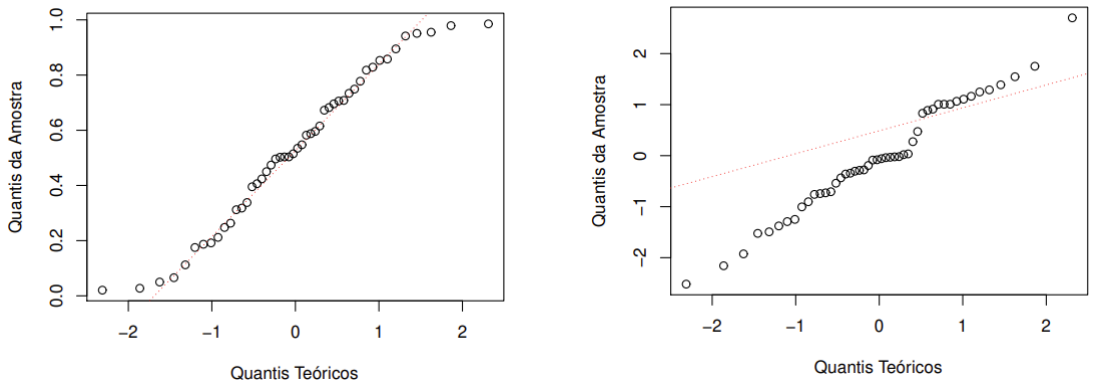
Para que um modelo de regressão linear seja considerado adequado, é necessário que os erros experimentais (resíduos) sejam Normais, Independentes e Identicamente Distribuídos (NIID), com média $\mu = 0$ e variância σ^2. Ou seja, variáveis randômicas, não relacionadas, aproximadas de uma distribuição Normal, apresentando independência em todos os níveis para os fatores (RODRIGUES, 2020). A verificação da aderência a estes requisitos pode ser dada tanto através da análise dos gráficos de pontos discrepantes e de distribuição dos erros quanto por testes estatísticos formais de normalidade e homogeneidade de variância (BROWN; FORSYTHE, 1974; MASSEY, 1951; SHAPIRO; WILK, 1965).

O gráfico **quantil-quantil da distribuição dos resíduos** compara quantis teóricos de distribuição Normal com os quantis dos resíduos dos experimentos. Com ele, é possível verificar se as amostras do estudo seguem ou se aproximam de uma distribuição Normal. A Figura 5 exemplifica esta ferramenta. Na Figura 5a é possível ver que os resíduos seguem a linha dos quantis teóricos, ou seja, apresentam distribuição Normal, enquanto na Figura 5b não se observa o mesmo fenômeno.

Figura 5 – Exemplos de gráficos quantil-quantil

(a) Distribuição Normal

(b) Não há aderência à distribuição Normal



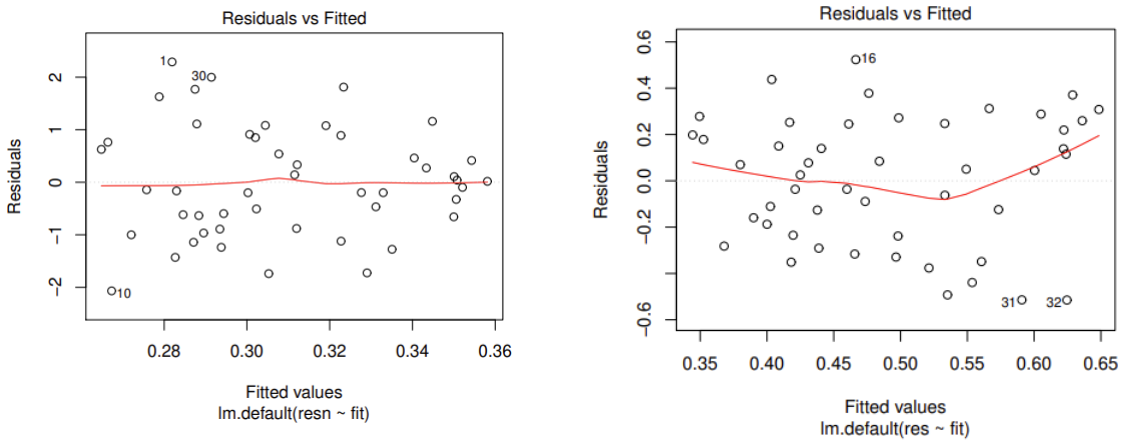
Fonte: Rodrigues (2020)

O gráfico de pontos discrepantes pode ser usado para verificar a propriedade da independência de resíduos. Nessa análise, deve-se verificar se os pontos dos resíduos estão distribuídos homogeneamente pelo gráfico, como o caso da Figura 6a, se não for o caso então existem indícios que a variância dos erros é uma função da variável de resposta e, nesse caso, os resíduos variam de forma heterogênea (como é o caso da Figura 6b) (RODRIGUES, 2020).

Figura 6 – Exemplo de gráfico de pontos discrepantes

(a) Homogeneidade de variância.

(b) Heterogeneidade de variância.

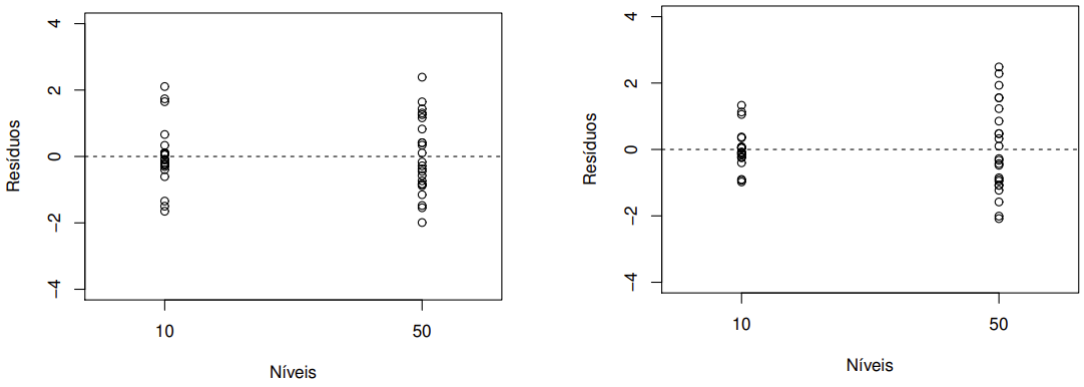


Fonte: Rodrigues (2020)

Para verificar a distribuição idêntica dos erros, pode-se usar do gráfico de Resíduos por Variável Regressora. Aqui, espera-se que os resíduos se apresentem igualmente distri-

Figura 7 – Resíduos por Variável Regressora

(a) Homogeneidade de variância. **(b)** Variação divergente entre níveis.



Fonte: Rodrigues (2020)

buídos formando linhas ortogonais ao eixo da abcissas de forma semelhante para os dois níveis de determinado fator, como mostra a Figura 7a. Outro tipo de distribuição, como a forma cônica presente na Figura 7b, indica ausência de distribuição idêntica.

Quando há divergência quanto a esses requisitos, uma solução possível de ser empregada é a transformação de dados. Para isso, Box e Cox 1964 propuseram a transformação de potência, que consiste em aplicar uma transformação y^{λ} nos dados inadequados. Após isso, a análise gráfica deve ser executada novamente.

2.5 SÍNTESE

Nessa seção foi definido o conceito de *Big Data*, bem como foram descritas as ferramentas para análise desse tipo de conjunto de dados. Após isso, foram descritos conceitos, funcionamento e arquitetura do *framework Spark*. Em seguida, foram apresentados os conceitos teóricos do delineamento experimental, método que servirá para a condução dos experimentos e análise dos resultados coletados.

3 TRABALHOS RELACIONADOS

Nesse capítulo são descritos, em termos gerais, trabalhos que se relacionam com o tema desse estudo. Os artigos e teses aqui mencionados propõem métodos para medir a influência de parâmetros *Spark* na sua performance além de também propor um método para configurar o *framework* para obter melhor eficiência (em termos de tempo). Entretanto, nenhum deles usa do delinamento experimental com essa finalidade. Há, porém, autores que usam do DoE, nesse caso, com a finalidade de analisar a influência de parâmetros de *hardware* e hiperparâmetros de algoritmos de aprendizagem de máquina.

3.1 OTIMIZAÇÃO DE PERFORMANCE DO *SPARK* COM AJUSTE DE PARÂMETROS DE CONFIGURAÇÃO

3.1.1 Método da tentativa e erro

Petridis, Gounaris e Torres (2017), por exemplo, desenvolveram uma metodologia com base na tentativa e falha para testar a performance de 12 parâmetros Spark, escolhidos entre mais de 150 com base no conhecimento e experiências dos autores. Nessa metodologia, os parâmetros estudados são testados individualmente para três tipos de aplicação: *sort-by-key*, *shuffling* e *K-Means*. O impacto no tempo de execução para cada teste foi analisado e foi obtida uma metodologia sequencial para ajuste de parâmetros. Então, essa metodologia foi testada para três diferentes tipos de aplicação, comparando o tempo de execução da tarefa com a configuração padrão e com a configuração obtida com o ajuste. Para a aplicação de *sort-by-key*, foi observado uma melhora de 44% na performance (de 218 segundos para 120 segundos). Para os testes de *K-Means*, foi observado uma melhora de 91% (654 segundos para 54 segundos). E, por fim, para os testes de *aggregate-by-key*, a melhora foi de 21%. No geral, conseguiu-se diminiuir o tempo de execução em mais de 10 vezes.

Gounaris e Torres (2018) aperfeiçoaram tal metodologia ao medir, também, o impacto de pares de parâmetros, além de aumentar a paralelização com a qual os testes podem ser conduzidos. A metodologia de ajuste de parâmetros proposta aqui conseguiu diminuir o tempo de execução em pelo menos 20%.

Usando também da metodologia de tentativa e erro, Ahmed et al. (2020) conduziram uma análise de 18 valores de parâmetros diferentes usando duas cargas de trabalho: *WordCount* e *TeraSort*. O trabalho mostra que a depender do tipo de aplicação e do tamanho da massa de dados, o ajuste de parâmetros pode levar a uma melhor performance. Por exemplo, para a carga de trabalho *WordCount*, o melhor valor para o parâmetro de paralelismo, com um aumento de performance de 3% é 300, e isso é evidenciado com

tamanho dos dados maior que 400GB.

3.1.2 Uso de aprendizagem de máquina

Alguns autores usaram algoritmos de *machine learning* em suas metodologias. Wang, Xu e He (2016) obtiveram um método usando modelos preditivos para estimar com qual configuração usando 13 parâmetros *Spark* pode se obter melhor performance. Os algoritmos utilizados para a predição foram o Árvore de Decisão, Regressão Logística, SVM e ANN. As cargas de trabalho analisadas foram *WordCount*, *Sort*, *Grep* e *Naïve Bayes*. O algoritmo de árvore de decisão se mostrou o melhor algoritmo para a escolha e ajustes dos parâmetros. Com isso, para validar a metodologia proposta, para cada benchmark foram comparados os tempos de execução com a configuração padrão *Spark* e aquela obtida com auxílio do modelo de aprendizagem de máquina. Por fim, foi possível observar que a metodologia proposta conseguiu melhorar a performance em até 55% e essa melhora é evidenciada a crescer do tamanho de dados de entrada.

Também usando algoritmos de aprendizagem de máquina, Nguyen, Khan e Wang (2018) estimaram o tempo de execução de jobs *Spark* para diferentes parâmetros e valores, os quais foram escolhidos com base em algoritmos de *feature selection*. Usando um *cluster* com 6 nós, 12 núcleos de CPU, 32GB de RAM e 1.8TB de disco rígido cada nó e utilizando 9 aplicações diferentes, dentre elas o *WordCount* e *TeraSort*, foram gerados dados com base nas combinações obtidas com o algoritmo de *feature selection* e, então, os algoritmos de aprendizagem de dados foram treinados com base nesses dados. Para avaliação da metodologia proposta, foi comparado a performance do sistema com configuração padrão e aquela obtida com o ajuste. Dessa forma, foi vista uma melhora de até 40% na performance em termos de tempo. Esse trabalho evidencia, mais uma vez, a diferença no ajuste de parâmetros a depender da aplicação em questão.

3.1.3 Uso de simulador

Chen et al. (2016) obteve um simulador na ferramenta *Intel®CoFluent™Studio*, usado para simular um *cluster Spark* e medir o tempo de execução com o dimensionamento de 33 parâmetros de *software* e 5 grupos de parâmetros de *hardware* diferentes. Porém, esse método exigiu conhecimento profundo da arquitetura de um *cluster Spark* para construção do simulador. Com base no simulador obtido, foi possível ajustar os parâmetros de configuração para melhora de 71% na performance de tarefas PageRank em comparação com os parâmetros padrões do Spark.

3.2 USO DE DELINEAMENTO EXPERIMENTAL

3.2.1 Aplicação de delineamento experimental e eliminação passo atrás

Para analisar os efeitos de fatores na qualidade de impressões 3D, Mohamed et al. (2016) usaram o delineamento fatorial para planejamento experimental, bem como da técnica de eliminação passo atrás para obtenção de modelos de regressão linear. Com essa metodologia, os autores conseguiram destacar parâmetros de maior impacto e, além disso, com os modelos de regressão, conseguiram determinar as condições ideais do processo para evitar problemas na qualidade do material.

Outro exemplo de emprego de tais técnicas pode ser visto no trabalho de Beltramino et al. (2016). Aplicando o método do delineamento fatorial completo para condução dos experimentos e o método da eliminação passo atrás para a obtenção de modelos de regressão linear, os autores conseguiram otimizar o processo de obtenção de nanocelulose levando em conta os fatores da quantidade de celulase, tempo de hidrólise ácida e temperatura de hidrólise ácida.

3.2.2 Delineamento experimental na computação

No trabalho de Lujan-Moreno et al. (2018), foi possível identificar os hiperparâmetros de maior impacto do algoritmo de Floresta Aleatória do *software* R. Os autores empregaram o delineamento fatorial fracionado para triagem dos fatores mais importantes e, depois, o fatorial completo para o ajuste de um modelo de segunda ordem usando técnicas de *Response-Surface Methodology* (RSM), técnica que também faz parte da metodologia de delineamento experimental. Aqui, a métrica de performance considerada foi a de acurácia balanceada. Com isso, o processo de otimização levou a uma melhora na performance do algoritmo, saindo de um valor de 0,6367 com a configuração padrão para 0,811 para a configuração obtida com a metodologia.

No contexto Spark, Rodrigues (2020) usou do delineamento experimental, mais precisamente o delineamento 2^k fatorial completo e fracionado, para investigar e ranquear parâmetros de *hardware* na execução em *cluster* de algoritmos de aprendizagem de máquina sobre um conjunto de dados *Big Data*. O tamanho do conjunto de dados também foi considerado na análise. Neste trabalho, evidenciou-se que o número de nós do *cluster*, tem um grande impacto no tempo de execução e, além disso, uma vez fornecida a quantidade mínima de memória para executar o algoritmo, sua mudança não afetou o tempo. Ademais, o autor obteve modelos de regressão linear para estimar o tempo e custo na execução desses algoritmos com base nos fatores em questão.

Este trabalho difere dos demais ao aplicar o delineamento experimental para analisar o impacto de parâmetros de *software* do *framework Spark*, obtendo um modelo de regressão linear com uso da eliminação passo atrás. Essa abordagem tem o intuito de auxiliar a

tomada de decisão testando as configurações de *software*, deixando as configurações de *hardware* fixas.

4 MATERIAIS E MÉTODOS

Esse capítulo tem como objetivo descrever os materiais utilizados no experimento, bem como os procedimentos seguidos no trabalho.

4.1 PLATAFORMA EXPERIMENTAL

4.1.1 *Hardware*

Para a primeira parte da pesquisa, foi simulado um *cluster* com 1 nó mestre e 3 trabalhadores em uma máquina com processador Intel®Core™i7-4790 3.6GHz com 8 núcleos, 16GB de RAM e um HD de 1TB. Para construção do *cluster* virtualizado, foi utilizada a plataforma *Docker*[3], e cada nó do *cluster* tem seus recursos compartilhados com os outros nós.

A segunda parte do trabalho usou *clusters* provisionados pelo serviço *Google Cloud*[4], mais precisamente o serviço *Dataproc*[5]. *O cluster* usado também foi criado com 1 mestre e 3 nós trabalhadores, cada nó contendo 2 vCPUs e 8GB de RAM. Para o armazenamento, o mestre tem 100GB e cada trabalhador tem 200GB.

4.1.2 *Software*

A versão do *Apache Spark* utilizada foi a 3.3.0 junto com o *Hadoop*, para uso de suas bibliotecas de cliente para o HDFS e YARN. O treinamento do algoritmo de aprendizagem de máquina foi conduzido, sendo utilizado a biblioteca MLlib do *Spark*. Para a análise estatística, o *software* SAS[6] foi utilizado, além dele, o *software*[7] R e foi utilizado junto com a biblioteca FrF2 (GRöMPING, 2022) para geração do plano experimental usando da técnica de delineamento fatorial fracionado.

4.2 ESTUDO DE CASO

O experimento foi conduzido alterando os parâmetros de configuração *Spark* na execução de um algoritmo de multiclassificação sobre o conjunto de dados PT7 Web.

[3] https://www.docker.com/
[4] https://cloud.google.com/
[5] https://console.cloud.google.com/dataproc
[6] https://www.sas.com/pt_br/home.html
[7] https://www.r-project.org/

4.2.1 Conjunto de dados

O PT7 Web é um corpus anotado em língua portuguesa composto por amostras coletadas de setembro de 2018 a março de 2020. Essas amostras foram coletadas de sete países de língua portuguesa: Angola, Brasil, Portugal, Cabo Verde, Guiné-Bissau, Macau e Moçambique. Os dados vieram do *Common Crawl*, que é um conjunto de dados de domínio público de páginas Web de vários idiomas, nesse caso, eles foram filtrados para os idiomas das nacionalidades citadas (RODRIGUES; VASCONCELOS; MACIEL, 2020). Os dados estão dispostos em tabelas em 200 arquivos no formato *.parquet*. As tabelas estão estruturadas da seguinte forma:

- *label*, do tipo string. Seus valores podem ser 1 para documentos brasileiros e 0 para aqueles que não são;
- *url*, do tipo string. Contém a fonte do documento original;
- *digest*, do tipo string. É uma função de hash do conteúdo da página;
- *raw*, do tipo string. É o documento no formato original em HTML puro;
- *token*, do tipo vetor. Se trata do documento dividido em palavras;
- *filtered*, do tipo vetor. Os tokens processados, *stopwords*.

Para esse trabalho, utilizou-se 95 arquivos no formato .parquet somando 68,36GB em estado bruto. Após pré-processamento, os dados usados para treinamento de modelo de aprendizagem de máquina teve tamanho de 14,88GB. Além disso, a coluna da classe (*label*) deixa de ser binária e passa a ter valores multiclasse.

4.2.2 Algoritmo de aprendizagem de máquina para classificação multiclasse

Foi utilizado o algoritmo de *Naïve Bayes* para multiclasse treinado sobre o conjunto de dados supracitado.

4.2.2.1 Naïve Baiyes

Este algoritmo é baseado no teorema de Bayes (1763), que pode ser formulado pela equação 4.1. Ele tem suma importância na estatística inferencial e trabalha com probabilidade condicional (BERRAR, 2019). Além disso, considera que os eventos são independentes, cuja suposição é simplista quando considerado o mundo real. Mesmo assim, o classificador *Naïve Bayes* se mostra eficiente quando testado em dados reais (WITTEN; FRANK; HALL, 2011).

Supondo um conjunto de dados, onde cada instância é descrita por um conjunto de atributos e pertence a apenas uma classe:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{4.1}$$

$P(c|x)$ é a probabilidade a posteriori da classe c dado os atributos x. $P(x|c)$ é a probabilidade dos atributos, dado a classe. E, por fim, $P(c)$ e $P(x)$ são as probabilidades a priori da classe c e dos atributos x.

A classificação se baseia em:

1. Calcular as probabilidades a priori para cada atributo
2. Calcular $P(x|c)$ com base nas probabilidades a priori calculadas
3. Para cada classe, calcular $P(c|x)$ com base na equação 4.1
4. A classe com maior probabilidade é o resultado da classificação

O modelo foi treinado para classificar os documentos do conjunto de dados e decidir sua nacionalidade de origem. Esse procedimento foi repetido diversas vezes, para diferentes parâmetros *Spark*, seguindo o plano experimental fornecido pelo delineamento fatorial fracionado.

Ressalta-se, porém, que a otimização da performance do algoritmo de aprendizagem de máquina não está no escopo da pesquisa, tampouco a análise de suas métricas. Seu treinamento serve, aqui, como meio para análise da performance do *job Spark*.

4.2.3 Parâmetros de configuração

Na Tabela 2 podem ser encontrados os parâmetros escolhidos, bem como seus níveis. Eles foram escolhidos baseados nas fontes descritas na Seção 2.2.4. Nesse caso, os parâmetros envolvendo configurações de *hardware* (exemplo: *spark.executor.memory, spark.executor.cores e spark.driver.memory*) foram deixados de fora da análise, além deles, também foram retirados da análise os parâmetros que dependem da definição de outros, a exemplo do *spark.dynamicAllocation.enabled*.

Para a definição dos valores, foi necessário escolher um intervalo que viabilizasse a condução dos experimentos e que também possibilitasse a observação de diferença na resposta de interesse analisada, o tempo de execução. Para essa definição, foram consideradas as limitações de *hardware*, bem como os valores recomendados pela literatura. Todos os valores dados em unidade de medida de informação (*bytes*) foram descritos em KB, para padronização.

Para os valores *spark.shuffle.file.buffer* e *spark.io.compression.lz4.blockSize*, cujos valores padrões é 32KB, a recomendação é aumentar esse valor para mais de 512KB, então

Tabela 2 – Parâmetros de configuração investigados

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

Fonte: A Autora (2023).

os limites inferiores foram escolhidos para abaixo de 32KB e o limite superior foi escolhido para mais que o dobro de 512KB. Com isso, foi definido o intervalo de 16KB a 2000KB (DAMJI et al., 2020).

Para os valores de *spark.sql.shuffle.partitions* e *spark.broadcast.blockSize*, foi recomendado que os valores sejam baixos a depender das limitações do *cluster*. Nesse caso, o limite superior foi definido com o valor padrão e o limite inferior foi definido com um valor mais distante do superior (DAMJI et al., 2020; NGUYEN; KHAN; WANG, 2018). Para o *spark.default.parallelism*, Damji et al. (2020) recomenda um valor de no máximo 10 para a maioria dos *clusters*, então levando em conta a limitação do *cluster* usado no trabalho, os limites foram definidos como 2 a 6.

O *spark.sql.files.maxPartitionBytes* se mostrou flexível, podendo ter seu limite inferior sendo definido como quase dez vezes menor que seu padrão, e o limite superior sendo quase dez vezes maior. Já o *spark.reducer.maxSizeInFlight* teve seus limites definidos com base no trabalho de Nguyen, Khan e Wang (2018), Ahmed et al. (2020).

4.3 PROCEDIMENTO

Foi conduzida uma investigação exploratória a fim de se obter conhecimento das limitações associadas à plataforma experimental. Além disso, essa etapa também serviu para a seleção dos parâmetros de configuração considerados no experimento, bem como desenvolver habilidades no manejo das ferramentas utilizadas. Ela consistiu do treinamento do algoritmo de aprendizagem de máquina escolhido, sob diferentes configurações. Aqui, servindo de testes de viabilidade e para confirmação da escolha do algoritmo de aprendizagem de máquina, foram treinados os algoritmos de Regressão Logística multiclasse bem como o de Árvore de Decisão, além do algoritmo *Naïve Bayes* já mencionado. Essa etapa foi conduzida no ambiente *Docker* e usou planos fatoriais fracionados com três, seis e doze replicações. Os parâmetros *spark.executor.memory* e *spark.driver.memory* foram deixados de fora da análise, mas tiveram que ser estudados e definidos de modo que possibilitassem

a condução dos experimentos.

Partiu-se então para os experimentos conduzidos no serviço *Google Dataproc*. Nessa etapa, *clusters* foram criados com as especificações descritas na Subseção 4.1.1. As tarefas submetidas ao *cluster* consiste no treinamento e teste de algoritmo *Naïve Bayes*. O plano experimental utilizado foi o fatorial fracionado randomizado de resolução V e número de replicações igual a três (Anexo A). Com a execução dos experimentos seguindo o plano obtido, o tempo de execução de cada *job* foi coletado.

Com posse dos resultados, foi então ajustado modelo de regressão linear, usando a técnica da eliminação passo atrás (descrita na Seção 2.4). Com isso, a influência de cada fator foi analisado e, por fim, as propriedades NIID dos resíduos do modelo foram verificadas graficamente.

5 RESULTADOS

Esse capítulo mostra e descreve os resultados obtidos a partir da condução das 192 unidades experimentais, mostradas no anexo A, na plataforma *Google Cloud Dataproc*. Além disso, a análise gráfica dos resíduos também é discutida.

5.1 PARÂMETROS IMPORTANTES E MODELO DE REGRESSÃO LINEAR

Tabela 3 – Projeto experimental $2_V^{7-1} \times 3$ para tempo NB (continua)

#	Plano	X1 (KB)	X2 (KB)	X3 (KB)	X4	X5 (KB)	X6	X7 (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
6	20-113-192	2000 $^+$	16 $^-$	1000000 $^+$	8 $^-$	24000 $^+$	2 $^-$	4000 $^+$	(285.32,302.54,294.20)	294.02
7	43-100-157	16 $^-$	2000 $^+$	1000000 $^+$	8 $^-$	24000 $^+$	2 $^-$	4000 $^+$	(272.86,307.86,283.22)	287.98
8	37-99-185	2000 $^+$	2000 $^+$	1000000 $^+$	8 $^-$	24000 $^+$	2 $^-$	16 $^-$	(278.67,285.31,290.03)	284.67
9	41-108-166	16 $^-$	16 $^-$	16000 $^-$	200 $^+$	24000 $^+$	2 $^-$	16 $^-$	(277.84,305.52,285.08)	289.48
10	12-122-154	2000 $^+$	16 $^-$	16000 $^-$	200 $^+$	24000 $^+$	2 $^-$	4000 $^+$	(301.14,292.76,285.67)	293.19
11	59-118-177	16 $^-$	2000 $^+$	16000 $^-$	200 $^+$	24000 $^+$	2 $^-$	4000 $^+$	(292.25,287.57,296.07)	291.97
12	56-73-152	2000 $^+$	2000 $^+$	16000 $^-$	200 $^+$	24000 $^+$	2 $^-$	16 $^-$	(283.81,277.31,291.73)	284.28
13	13-104-180	16 $^-$	16 $^-$	1000000 $^+$	200 $^+$	24000 $^+$	2 $^-$	4000 $^+$	(278.96,301.86,286.54)	289.12
14	35-86-158	2000 $^+$	16 $^-$	1000000 $^+$	200 $^+$	24000 $^+$	2 $^-$	16 $^-$	(271.96,279.13,292.21)	281.10
15	40-85-156	16 $^-$	2000 $^+$	1000000 $^+$	200 $^+$	24000 $^+$	2 $^-$	16 $^-$	(298.78,280.52,292.18)	290.50
16	55-107-165	2000 $^+$	2000 $^+$	1000000 $^+$	200 $^+$	24000 $^+$	2 $^-$	4000 $^+$	(282.39,291.24,298.17)	290.60
17	44-71-146	16 $^-$	16 $^-$	16000 $^-$	8 $^-$	96000 $^+$	2 $^-$	16 $^-$	(272.84,296.53,283.31)	284.23
18	64-101-163	2000 $^+$	16 $^-$	16000 $^-$	8 $^-$	96000 $^+$	2 $^-$	4000 $^+$	(283.62,313.98,288.17)	295.26
19	26-79-129	16 $^-$	2000 $^+$	16000 $^-$	8 $^-$	96000 $^+$	2 $^-$	4000 $^+$	(264.83,269.64,281.19)	271.89
10	1-102-186	2000 $^+$	2000 $^+$	16000 $^-$	8 $^-$	96000 $^+$	2 $^-$	16 $^-$	(273.25,317.16,271.36)	287.26
21	36-105-140	16 $^-$	16 $^-$	1000000 $^+$	8 $^-$	96000 $^+$	2 $^-$	4000 $^+$	(274.24,324.42,288.39)	295.68
22	54-123-145	2000 $^+$	16 $^-$	1000000 $^+$	8 $^-$	96000 $^+$	2 $^-$	16 $^-$	(287.21,289.93,290.67)	289.27
23	38-119-184	16 $^-$	2000 $^+$	1000000 $^+$	8 $^-$	96000 $^+$	2 $^-$	16 $^-$	(280.27,307.23,309.65)	299.05
24	61-91-172	2000 $^+$	2000 $^+$	1000000 $^+$	8 $^-$	96000 $^+$	2 $^-$	4000 $^+$	(286.72,290.81,298.11)	291.88
25	10-125-155	16 $^-$	16 $^-$	16000 $^-$	200 $^+$	96000 $^+$	2 $^-$	4000 $^+$	(276.32,292.21,288.69)	285.74
26	63-110-132	2000 $^+$	16 $^-$	16000 $^-$	200 $^+$	96000 $^+$	2 $^-$	16 $^-$	(286.28,297.87,282.74)	288.96
27	21-68-144	16 $^-$	2000 $^+$	16000 $^-$	200 $^+$	96000 $^+$	2 $^-$	16 $^-$	(272.84,288.80,301.26)	287.63
28	14-78-133	2000 $^+$	2000 $^+$	16000 $^-$	200 $^+$	96000 $^+$	2 $^-$	4000 $^+$	(277.62,284.69,282.09)	281.46
29	47-88-171	16 $^-$	16 $^-$	1000000 $^+$	200 $^+$	96000 $^+$	2 $^-$	16 $^-$	(284.26,297.10,284.45)	288.61
30	51-127-164	2000 $^+$	16 $^-$	1000000 $^+$	200 $^+$	96000 $^+$	2 $^-$	4000 $^+$	(305.69,295.18,297.74)	299.56
31	2-94-179	16 $^-$	2000 $^+$	1000000 $^+$	200 $^+$	96000 $^+$	2 $^-$	4000 $^+$	(285.02,334.01,290.99)	303.34
32	60-82-136	2000 $^+$	2000 $^+$	1000000 $^+$	200 $^+$	96000 $^+$	2 $^-$	16 $^-$	(292.53,290.03,281.82)	288.13

Fonte: A Autora (2023).

As Tabelas 3 e 4 mostram o tempo de execução (em segundos) para os experimentos conduzidos baseado no plano 2_V^{7-1} com três replicações[8]. O número de unidades experimentais executadas foi de $2^6 \times 3 = 192$. A primeira coluna mostra o experimento, já a segunda coluna mostra a ordem com a qual o experimento foi conduzido (levando em conta, também, suas replicações). Por exemplo, o experimento #1 teve três replicações, estas executadas na 53^a, 103^a e 159^a posição do projeto. A seguir, tem-se as colunas dos parâmetros de configuração e, depois, a coluna com os resultados obtidos com a execução

[8] Um projeto de experimento 2_V^{7-1} significa um projeto fatorial completo 2^7 reduzido para um fracionado 2^6 com resolução V.

Tabela 4 – Projeto experimental $2_V^{7-1} \times 3$ para tempo NB (conclusão)

#	Plano	X1 (KB)	X2 (KB)	X3 (KB)	X4	X5 (KB)	X6	X7 (KB)	t_1, t_2, t_3 (s)	$\bar{t}$
33	52-93-147	16 $^-$	16 $^-$	16000 $^-$	8 $^-$	24000 $^{+-}$	6 $^+$	16 $^-$	(292.79,296.57,276.30)	288.56
34	49-97-191	2000 $^+$	16 $^-$	16000 $^-$	8 $^-$	24000 $^{+-}$	6 $^+$	4000 $^+$	(285.79,298.32,279.06)	287.72
35	22-69-183	16 $^-$	2000 $^+$	16000 $^-$	8 $^-$	24000 $^{+-}$	6 $^+$	4000 $^+$	(285.05,297.22,288.81)	290.36
36	57-109-169	2000 $^+$	2000 $^+$	16000 $^-$	8 $^-$	24000 $^{+-}$	6 $^+$	16 $^-$	(277.66,290.39,279.94)	282.66
37	3-128-138	16 $^-$	16 $^-$	1000000 $^+$	8 $^-$	24000 $^{+-}$	6 $^+$	4000 $^+$	(287.53,286.17,282.40)	285.37
38	27-112-175	2000 $^+$	16 $^-$	1000000 $^+$	8 $^-$	24000 $^{+-}$	6 $^+$	16 $^-$	(297.83,298.45,273.45)	289.91
39	28-92-131	16 $^-$	2000 $^+$	1000000 $^+$	8 $^-$	24000 $^{+-}$	6 $^+$	16 $^-$	(292.22,289.63,301.74)	294.53
40	6-120-190	2000 $^+$	2000 $^+$	1000000 $^+$	8 $^-$	24000 $^{+-}$	6 $^+$	4000 $^+$	(281.42,295.86,292.48)	289.92
41	58-83-162	16 $^-$	16 $^-$	16000 $^-$	200 $^+$	24000 $^{+-}$	6 $^+$	4000 $^+$	(292.18,294.39,294.10)	293.56
42	50-80-141	2000 $^+$	16 $^-$	16000 $^-$	200 $^+$	24000 $^{+-}$	6 $^+$	16 $^-$	(302.37,303.28,280.39)	295.34
43	33-70-168	16 $^-$	2000 $^+$	16000 $^-$	200 $^+$	24000 $^{+-}$	6 $^+$	16 $^-$	(289.53,287.52,298.50)	291.85
44	19-75-137	2000 $^+$	2000 $^+$	16000 $^-$	200 $^+$	24000 $^{+-}$	6 $^+$	4000 $^+$	(282.11,285.44,274.24)	280.60
45	18-95-161	16 $^-$	16 $^-$	1000000 $^+$	200 $^+$	24000 $^{+-}$	6 $^+$	16 $^-$	(290.43,297.91,298.23)	295.52
46	4-98-150	2000 $^+$	16 $^-$	1000000 $^+$	200 $^+$	24000 $^{+-}$	6 $^+$	4000 $^+$	(287.42,342.89,288.88)	306.39
47	48-65-189	16 $^-$	2000 $^+$	1000000 $^+$	200 $^+$	24000 $^{+-}$	6 $^+$	4000 $^+$	(285.18,293.77,279.41)	286.12
48	8-84-176	2000 $^+$	2000 $^+$	1000000 $^+$	200 $^+$	24000 $^{+-}$	6 $^+$	16 $^-$	(299.75,292.31,281.83)	291.30
49	25-96-151	16 $^-$	16 $^-$	16000 $^-$	8 $^-$	96000 $^+$	6 $^+$	4000 $^+$	(272.39,310.39,282.86)	288.55
50	15-121-174	2000 $^+$	16 $^-$	16000 $^-$	8 $^-$	96000 $^+$	6 $^+$	16 $^-$	(267.92,286.62,277.95)	277.50
51	34-90-142	16 $^-$	2000 $^+$	16000 $^-$	8 $^-$	96000 $^+$	6 $^+$	16 $^-$	(284.92,293.47,287.49)	288.63
52	17-81-173	2000 $^+$	2000 $^+$	16000 $^-$	8 $^-$	96000 $^+$	6 $^+$	4000 $^+$	(277.56,287.32,280.75)	281.88
53	30-115-181	16 $^-$	16 $^-$	1000000 $^+$	8 $^-$	96000 $^+$	6 $^+$	16 $^-$	(282.70,299.05,311.07)	297.61
54	42-106-187	2000 $^+$	16 $^-$	1000000 $^+$	8 $^-$	96000 $^+$	6 $^+$	4000 $^+$	(286.08,288.73,281.97)	285.59
55	9-116-170	16 $^-$	2000 $^+$	1000000 $^+$	8 $^-$	96000 $^+$	6 $^+$	4000 $^+$	(274.09,285.64,291.45)	283.73
56	11-66-178	2000 $^+$	2000 $^+$	1000000 $^+$	8 $^-$	96000 $^+$	6 $^+$	16 $^-$	(284.41,295.87,287.10)	289.13
57	29-72-182	16 $^-$	16 $^-$	16000 $^-$	200 $^+$	96000 $^+$	6 $^+$	16 $^-$	(287.87,276.82,279.88)	281.52
58	62-67-153	2000 $^+$	16 $^-$	16000 $^-$	200 $^+$	96000 $^+$	6 $^+$	4000 $^+$	(291.33,284.56,283.14)	286.35
59	46-124-135	16 $^-$	2000 $^+$	16000 $^-$	200 $^+$	96000 $^+$	6 $^+$	4000 $^+$	(281.29,276.11,277.09)	278.16
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

Fonte: A Autora (2023).

Tabela 5 – Representação em símbolo dos parâmetros de configuração Spark

Nome do parâmetro	Símbolo
spark.shuffle.file.buffer	X1
spark.io.compression.lz4.blockSize	X2
spark.sql.files.maxPartitionBytes	X3
spark.sql.shuffle.partitions	X4
spark.reducer.maxSizeInFlight	X5
spark.default.parallelism	X6
spark.broadcast.blockSize	X7

Fonte: A Autora (2023).

do experimento, considerando o valor das três replicações. Por último, tem-se a coluna da média dos resultados, para cada experimento. Os parâmetros de configuração analisados estão representados simbolicamente conforme mostra a Tabela 5.

A Tabela 6 mostra os parâmetros presentes no modelo final obtido pelo método da eliminação passo atrás, ou seja, as variáveis de maior importância para a resposta de saída. Nota-se que o parâmetro *spark.sql.shuffle.partitions* não está presente no modelo. O ponto de corte do nível de significância foi de 5%, e todas as variáveis do modelo se mostram significativas, pois têm o $p\text{-}value < 0,05$. Na coluna de estimativa de parâmetro é possível

ver os coeficientes para cada variável, os coeficientes positivos indicam impacto positivo na resposta analisada, os negativos indicam impacto negativo. Ou seja quanto maior o valor dos parâmetros com coeficiente positivo maior o tempo de execução de tarefa. Para os coeficientes negativos, quanto maior o valor do parâmetro, menor é o tempo de execução.

Os parâmetros destacados são:

- *spark.default.parallelism*, com impacto positivo.
- Interação *spark.shuffle.file.buffer* com *spark.default.parallelism*, com impacto negativo
- Interação *spark.shuffle.file.buffer* com *spark.broadcast.blockSize*, com impacto positivo.
- Interação *spark.io.compression.lz4.blockSize* com *spark.broadcast.blockSize*, com impacto negativo
- Interação *spark.sql.files.maxPartitionBytes* com *spark.reducer.maxSizeInFlight*, com impacto positivo.
- Interação *spark.reducer.maxSizeInFlight* com *spark.default.parallelism*, com impacto negativo

Tabela 6 – Estimativas de parâmetro

Variável	GL	Estimativa de parâmetro	Erro padrão	t-value	p-value
Intercept	1	$286,49148$	1,93574	148.00	<.0001
X6	1	$1,33830$	0,53994	2.48	0.0141
X1X6	1	$-0,00045818$	0.00020747	-2.21	0.0284
X1X7	1	$6,625831 * 10^{7}$	$2.835216 * 10^{7}$	2.34	0.0205
X2X7	1	$-6,52875 * 10^{-7}$	$2.427972 * 10^{7}$	-2.69	0.0078
X3X5	1	$8,21679 * 10^{-11}$	$2.17448 * 10^{-11}$	3.78	0.0002
X5X6	1	-0,00001706	0.00000527	-3.24	0.0014

Fonte: A Autora (2023).

Com os coeficientes obtidos com o modelo, é possível construir a Equação reduzida 5.1

$$t = 286,49148 + 1,33830X1 - 0,00045818X1X6 + 6,625831 \times 10^{-6}X1X7 - 6,52875 \times 10^{-7}X2X7 + 8,21679 \times 10^{-11}X3X5 - 0,00001706X5X6 \quad (5.1)$$

As informações do modelo obtido são mostrados nas Tabelas 7 e 8. O modelo tem $R^2 = 0,1320$ e $R^2_{ajustado} = 0,1038$, valores aquém do satisfatório para previsão precisa da variável dependente. Porém, se mostra significativo com $p\text{-}value = 0,0002 < 0,05$.

Com os resultados obtidos, e possível notar, por exemplo, que para o treinamento do algoritmo Naïve Bayes, no *cluster* Google Cloud especificado no capítulo 4, aumentar o grau de paralelismo leva a uma piora do desempenho *Spark*. Nesse caso, é importante ressaltar que, apesar de os resultados servirem para obter um entendimento geral do funcionamento *Spark*, eles estão restritos à aplicação e contexto com a qual os experimentos foram testados (algoritmo de aprendizagem de máquina, volume de dados e especificações do *cluster*).

Tabela 7 – 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			

Fonte: A Autora (2023).

Tabela 8 – 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		

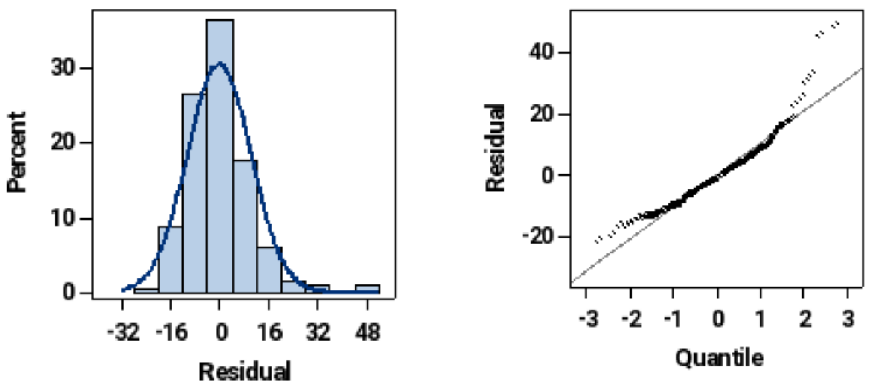
Fonte: A Autora (2023).

5.2 ANÁLISE DOS RESÍDUOS

Com o modelo de regressão linear obtido, é necessário então analisar os resíduos para investigar a adequação do modelo às suposições das propriedades NIID, descritas na Seção 2.4.1.

Figura 8 – Normalidade dos resíduos

(a) Histograma dos resíduos **(b)** Quantil-quantil

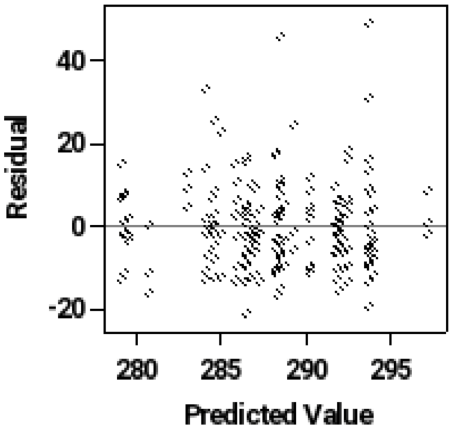


Fonte: A Autora (2023).

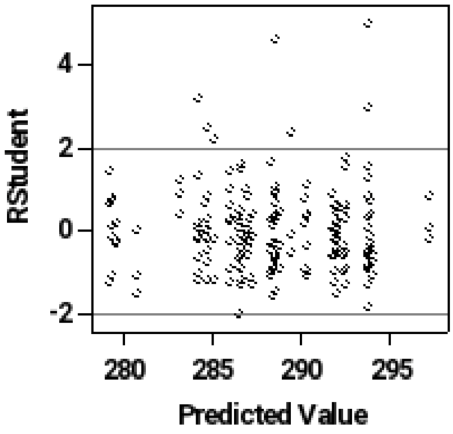
A Figura 8a mostra o histograma dos resíduos e a Figura 8b mostra o gráfico quantil-quantil de distribuição de resíduos. É notável o formato de distribuição Normal do gráfico do histograma. Para o gráfico quantil-quantil, existem pontos afastados da reta considerados *outliers*, porém a maioria dos pontos está concentrado sobre a reta dos quantis teóricos, o que evidencia a normalidade dos resíduos.

Figura 9 – Gráficos de Resíduos *versus* Valores ajustados

(a) Resíduos *versus* Valores ajustados

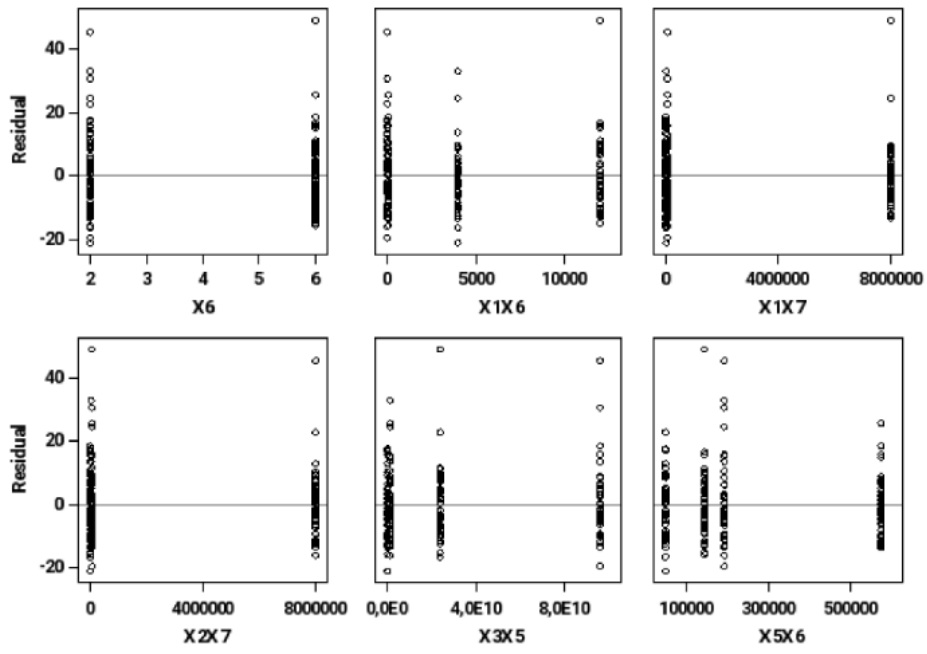


(b) Resíduos estudentizados *versus* Valores ajustados



Fonte: A Autora (2023).

Figura 10 – Gráficos de Resíduos *versus* Variáveis regressoras



Fonte: A Autora (2023).

Além disso, a Figura 9 mostram 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, há o gráfico de Resíduos versus Regressores presente na Figura 10. Nele, é possível ver os resíduos igualmente distribuídos. O gráfico com os regressores X1X7 apresenta diferença na distribuição, porém ainda aceitável para uso do modelo.

Com isso, verifica-se a adequação do modelo no que se refere às suposições básicas de normalidade, com base na análise de seus resíduos.

6 CONSIDERAÇÕES FINAIS

Com o crescer da necessidade de análise *Big Data*, vem à tona a importância em otimizar tarefas, principalmente, para poupar recursos. Esse trabalho tem como objetivo identificar parâmetros de configuração *Spark* que mais impactam o tempo de execução de tarefas.

Com uso de técnicas de delineamento experimental, foi possível diminuir o número de experimentos necessários para tirar conclusões sobre a importância dos fatores analisados na pesquisa, visto que um modelo de regressão linear considerando os efeitos principais e interações de segunda ordem foi obtido a partir dos resultados dos experimentos.

Apesar do modelo ter valores de R^2 e $R^2_{ajustado}$ aquém do considerado satisfatório e não ser confiável para para se obter uma estimativa precisa do tempo de execução de acordo com os valores de suas variáveis regressoras, o modelo de regressão linear mostra-se significativo, com $p\text{-}value < 0,05$, o que significa que ele mostra o modo com o qual o tempo de execução varia de acordo com suas variáveis regressoras. Além disso, ele também indica com significância, os fatores e interações de maior importância para variação da resposta analisada (MINITAB, 2019). Por exemplo, é possível concluir que um aumento no *spark.default.parallelism* acarraterá em um aumento do tempo de execução. Ao contrário do produto do valor do *spark.shuffle.file.buffer* e *spark.default.parallelism*, cujo aumento leva a uma diminuição do tempo de execução da tarefa.

É importante ressaltar que os resultados obtidos têm valor no contexto específico da pesquisa. Isso porque, há evidências que, ao mudar de aplicação, volume e natureza dos dados e plataforma experimental, principalmente *hardware*, os parâmetros de maior importância para variação do tempo de execução também mudam, como mostrado em trabalhos relacionados descritos no Capítulo 3.

Nesse contexto, a pesquisa teve como resultados:

- A implementação de tarefa sobre a base de dados PT7 Web
- Identificação de parâmetros mais importantes, dado o contexto
- Obtenção de modelo de regressão linear que auxilia no entendimento da variação do tempo de execução com base nos parâmetros de configuração de maior importância, apesar de não a estimar com precisão

6.1 DIFICULDADES ENCONTRADAS

A principal dificuldade encontrada na pesquisa se refere à obtenção de modelo que se adequasse à premissa da normalidade, visto que, com a investigação exploratória e execução de experimentos com três, seis e doze replicações em plataforma *Docker*, não foi possível obter nenhum modelo cujos resíduos tivessem distribuição Normal. Essa dificuldade indica a possibilidade de que a plataforma experimental referente à essa etapa do trabalho tivessem variáveis de fundo o suficiente para prejudicar a obtenção de resultados adequados.

A obtenção de recursos computacionais para possibilitar a execução da pesquisa também foi um fator de limitação, visto que a documentação do *Spark* sugere oito a dezesseis núcleos por nó e o utilizado na pesquisa foi de dois núcleos por nó. O impacto dessa limitação, porém, não pôde ser medido.

6.2 TRABALHOS FUTUROS

Como oportunidades para trabalhos futuros, destacam-se as opções descritas abaixo.

Analisar mais ou diferentes parâmetros de configuração daqueles analisados nessa pesquisa. Nesse caso, pode-se, também, combinar parâmetros de *software* com fatores de *hardware*, a exemplo do número de núcleos, tamanho da memória RAM, entre outros. Ademais, é possível também a análise de outra variável dependente podendo ser, por exemplo: uso de disco, volume de tráfego de rede ou sobrecarga de CPU.

Além disso, pode-se, também, empregar técnicas para adição de pontos axiais e centrais no plano experimental a fim de se detectar uma relação não linear entres as variáveis independentes e a variável dependente. Como exemplo, pode-se citar o delineamento composto central e o delineamento Box-Behnken (MONTGOMERY, 2013). A partir disso, é possível aplicar a metodologia da superfície de resposta para otimização da saída de interesse (MONTGOMERY, 2013).

Ainda como oportunidade de estender o presente trabalho, há o emprego de outras tarefas de análise *Big Data*. Estas podendo ser tarefas SQL, *Streaming* ou outros algoritmos de aprendizagem de máquina.

E, por fim, há a execução de experimentos em *clusters* maiores e mais poderosos, seguindo a recomendação da documentação *Spark*.

REFERÊNCIAS

AHMED, N. et al. A comprehensive performance analysis of apache hadoop and apache spark for large scale data sets using hibench. **J. Big Data**, v. 7, n. 1, p. 110, 2020. Disponível em: <https://doi.org/10.1186/s40537-020-00388-5>. Citado 4 vezes nas páginas 21, 22, 33 e 40.

AMATO, A. On the role of distributed computing in big data analytics. In: _____. **Distributed Computing in Big Data Analytics: Concepts, Technologies and Applications**. Cham: Springer International Publishing, 2017. p. 1–10. ISBN 978-3-319-59834-5. Disponível em: <https://doi.org/10.1007/978-3-319-59834-5_1>. Citado 3 vezes nas páginas 13, 16 e 17.

AMIN, M. M.; KIANI, A. Multi-disciplinary analysis of a strip stabilizer using body-fluid-structure interaction simulation and design of experiments (doe). **Journal of Applied Fluid Mechanics**, v. 13, n. 1, p. 261–273, 2020. ISSN 1735-3572. Disponível em: <https://www.jafmonline.net/article_917.html>. Citado na página 23.

BAYES, T. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. **Philosophical Transactions of the Royal Society of London**, v. 53, p. 370–418, 1763. Citado na página 38.

BELTRAMINO, F. et al. Optimization of sulfuric acid hydrolysis conditions for preparation of nanocrystalline cellulose from enzymatically pretreated fibers. **Cellulose**, v. 23, n. 3, p. 1777–1789, Mar 2016. Citado na página 35.

BERRAR, D. Bayes' theorem and naive bayes classifier. In: RANGANATHAN, S. et al. (Ed.). **Encyclopedia of Bioinformatics and Computational Biology**. Oxford: Academic Press, 2019. p. 403–412. ISBN 978-0-12-811432-2. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128096338204731>. Citado na página 38.

BOX, G.; COX, D. An analysis of transformations (with discussion). **Journal of the Royal Statistical Society, Series B**, v. 26, p. 211–252, 01 1964. Citado na página 32.

BOX, G. E. P.; MEYER, R. D. An analysis for unreplicated fractional factorials. **Technometrics**, v. 28, p. 11–18, 1986. Citado na página 23.

BOX, G. E. P.; WILSON, K. B. On the experimental attainment of optimum conditions. **Journal of the Royal Statistical Society: Series B (Methodological)**, v. 13, n. 1, p. 1–38, 1951. Disponível em: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1951.tb00067.x>. Citado na página 23.

BROWN, M. B.; FORSYTHE, A. B. Robust tests for the equality of variances. **Journal of the American Statistical Association**, [American Statistical Association, Taylor Francis, Ltd.], v. 69, n. 346, p. 364–367, 1974. ISSN 01621459. Disponível em: <http://www.jstor.org/stable/2285659>. Citado na página 30.

CASLER, M. D. Fundamentals of experimental design: Guidelines for designing successful experiments. **Agronomy Journal**, v. 107, n. 2, p. 692–705, 2015. Disponível em: <https://acsess.onlinelibrary.wiley.com/doi/abs/10.2134/agronj2013.0114>. Citado na página 23.

CHAMBERS, B.; ZAHARIA, M. **Spark: The Definitive Guide Big Data Processing Made Simple**. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2018. ISBN 1491912219. Citado 3 vezes nas páginas 18, 19 e 20.

CHARNET REINALDO; FREIRE, C. A. D. L. C. E. M. R. B. H. **Análise de modelos de regressão linear com aplicações -**. 1. ed. Campinas, SP: Editora da Unicamp, 1999. ISBN 978—8-5-26-80-4. Citado 2 vezes nas páginas 29 e 30.

CHEN, Q. et al. Simulating spark cluster for deployment planning, evaluation and optimization. In: **2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)**. [S.l.: s.n.], 2016. p. 1–11. Citado 3 vezes nas páginas 13, 21 e 34.

DAMJI, J. et al. **Learning Spark, 2nd Edition**. O'Reilly Media, Incorporated, 2020. ISBN 9781492050032. Disponível em: <https://pages.databricks.com/rs/094-YMS-629/images/LearningSpark2.0.pdf>. Citado 6 vezes nas páginas 18, 19, 20, 21, 22 e 40.

DASH, S. et al. Big data in healthcare: management, analysis and future prospects. **J. Big Data**, v. 6, p. 54, 2019. Disponível em: <https://doi.org/10.1186/s40537-019-0217-0>. Citado na página 13.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 51, n. 1, p. 107–113, jan 2008. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1327452.1327492>. Citado na página 13.

FISHER, R. A. **The Design of Experiments**. Edinburgh: Oliver and Boyd, 1935. Citado na página 23.

GHEMAWAT, S.; GOBIOFF, H.; LEUNG, S.-T. The google file system. In: **Proceedings of the 19th ACM Symposium on Operating Systems Principles**. Bolton Landing, NY: [s.n.], 2003. p. 20–43. Citado na página 13.

GOUNARIS, A.; TORRES, J. A methodology for spark parameter tuning. **Big Data Research**, v. 11, p. 22–32, 2018. ISSN 2214-5796. Selected papers from the 2nd INNS Conference on Big Data: Big Data Neural Networks. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214579617300114>. Citado 3 vezes nas páginas 20, 21 e 33.

GRöMPING, U. **Package FRF2**. Comprehensive R Archive Network (CRAN), 2022. Disponível em: <https://cran.r-project.org/web/packages/FrF2/index.html>. Citado na página 37.

GUECIOUER, D.; YOUCEF, G.; TAREK, N. Rheological and mechanical optimization of a steel fiber reinforced self-compacting concrete using the design of experiments method. **European Journal of Environmental and Civil Engineering**, Taylor Francis, v. 26, n. 3, p. 1097–1117, 2022. Disponível em: <https://doi.org/10.1080/19648189.2019.1697758>. Citado na página 23.

HASHEM, I. A. T. et al. The rise of “big data” on cloud computing: Review and open research issues. **Information Systems**, v. 47, p. 98–115, 2015. ISSN 0306-4379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306437914001288>. Citado 2 vezes nas páginas 13 e 16.

HUAI, Y. et al. Dot: A matrix model for analyzing, optimizing and deploying software for big data analytics in distributed systems. **Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC 2011**, 10 2011. Citado na página 17.

JIN, D.-H.; KIM, H.-J. Integrated understanding of big data, big data analysis, and business intelligence: A case study of logistics. **Sustainability**, v. 10, n. 10, 2018. ISSN 2071-1050. Disponível em: <https://www.mdpi.com/2071-1050/10/10/3778>. Citado na página 13.

KAMBATLA, K. et al. Trends in big data analytics. **Journal of Parallel and Distributed Computing**, v. 74, n. 7, p. 2561–2573, 2014. ISSN 0743-7315. Special Issue on Perspectives on Parallel and Distributed Processing. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731514000057>. Citado na página 13.

KUTNER, M. **Applied Linear Statistical Models**. McGraw-Hill Irwin, 2005. (McGrwa-Hill international edition). ISBN 9780071122214. Disponível em: <https://books.google.com.br/books?id=0xqCAAAACAAJ>. Citado na página 29.

LANEY, D. **3D Data Management: Controlling Data Volume, Velocity, and Variety**. [S.l.], 2001. Disponível em: <http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>. Citado na página 16.

LUJAN-MORENO, G. A. et al. Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. **Expert Systems with Applications**, v. 109, p. 195–205, 2018. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417418303178>. Citado 2 vezes nas páginas 23 e 35.

MASSEY, F. J. The kolmogorov-smirnov test for goodness of fit. **Journal of the American Statistical Association**, [American Statistical Association, Taylor Francis, Ltd.], v. 46, n. 253, p. 68–78, 1951. ISSN 01621459. Disponível em: <http://www.jstor.org/stable/2280095>. Citado na página 30.

MINITAB, E. **https://blog.minitab.com/pt/analise-de-regressao-como-interpretar-o-r-quadrado-e-avaliar-a-qualidade-de-ajuste**. 2019. Disponível em: <https://blog.minitab.com/pt/analise-de-regressao-como-interpretar-o-r-quadrado-e-avaliar-a-qualidade-de-ajuste>. Citado na página 48.

MISHRA, V. et al. Quality by design (qbd) approaches in current pharmaceutical set-up. **Expert Opinion on Drug Delivery**, Taylor Francis, v. 15, n. 8, p. 737–758, 2018. PMID: 30044646. Disponível em: <https://doi.org/10.1080/17425247.2018.1504768>. Citado na página 23.

MOHAMED, O. A. et al. Effect of process parameters on dynamic mechanical performance of fdm pc/abs printed parts through design of experiment. **Journal of Materials Engineering and Performance**, v. 25, n. 7, p. 2922–2935, Apr 2016. Citado na página 35.

MONTGOMERY, D.; RUNGER, G. **Estatística aplicada e probabilidade para engenheiros**. Livros Técnicos e Científicos, 2003. ISBN 9788521613602. Disponível em: <https://books.google.com.br/books?id=hkt0AAAACAAJ>. Citado na página 29.

MONTGOMERY, D. C. **Design and Analysis of Experiments**. 8. ed. New Jersey, U.S.A: WILEY, 2013. ISBN 978-1118-14692-7. Citado 5 vezes nas páginas 14, 22, 23, 25 e 49.

MYERS, R. H.; MONTGOMERY, D. C.; ANDERSON-COOK, C. M. **Response Surface Methodology**: Process and product optimization using designed experiments. 5. ed. New Jersey, U.S.A: WILEY, 2016. ISBN 978-1-118-91601-8. Citado 4 vezes nas páginas 24, 25, 26 e 28.

NGUYEN, N.; KHAN, M. M. H.; WANG, K. Towards automatic tuning of apache spark configuration. In: **2018 IEEE 11th International Conference on Cloud Computing (CLOUD)**. [S.l.: s.n.], 2018. p. 417–425. Citado 6 vezes nas páginas 14, 20, 21, 22, 34 e 40.

PACKIANATHER, M. S.; DRAKE, P. R.; ROWLANDS, H. Optimizing the parameters of multilayered feedforward neural networks through taguchi design of experiments. **Quality and Reliability Engineering International**, v. 16, n. 6, p. 461–473, 2000. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/1099-1638%28200011/12%2916%3A6%3C461%3A%3AAID-QRE341%3E3.0.CO%3B2-G>. Citado na página 23.

PETRIDIS, P.; GOUNARIS, A.; TORRES, J. Spark parameter tuning via trial-and-error. In: ANGELOV, P. et al. (Ed.). **Advances in Big Data**. Cham: Springer International Publishing, 2017. p. 226–237. ISBN 978-3-319-47898-2. Citado 2 vezes nas páginas 21 e 33.

POLITIS, S. N. et al. Design of experiments (doe) in pharmaceutical development. **Drug Development and Industrial Pharmacy**, Taylor Francis, v. 43, n. 6, p. 889–901, 2017. PMID: 28166428. Disponível em: <https://doi.org/10.1080/03639045.2017.1291672>. Citado na página 23.

RODRIGUES, J.; VASCONCELOS, G.; MACIEL, P. **PT7 Web, an Annotated Portuguese Language Corpus**. IEEE Dataport, 2020. Disponível em: <https://dx.doi.org/10.21227/fhrm-n966>. Citado 2 vezes nas páginas 14 e 38.

RODRIGUES, J.; VASCONCELOS, G.; MACIEL, P. Screening hardware and volume factors in distributed machine learning algorithms on spark: A design of experiments (doe) based approach. **Computing**, v. 103, 10 2021. Citado na página 14.

RODRIGUES, J. B. **Análise de Fatores Relevantes no Desempenho de Plataformas para Processamento de Big Data**: Uma abordagem baseada em projeto de experimentos. Tese (Doutorado), Recife, 2020. Disponível em: <https://repositorio.ufpe.br/handle/123456789/39207>. Citado 8 vezes nas páginas 13, 16, 23, 24, 30, 31, 32 e 35.

SALLOUM, S. et al. Big data analytics on apache spark. **International Journal of Data Science and Analytics**, v. 1, p. 145–164, 2015. ISSN 2364-4168. Citado 4 vezes nas páginas 17, 18, 19 e 20.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, [Oxford University Press, Biometrika Trust], v. 52, n. 3/4, p. 591–611, 1965. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333709>. Citado na página 30.

SIMONET, A.; FEDAK, G.; RIPEANU, M. Active data: A programming model to manage data life cycle across heterogeneous systems and infrastructures. **Future Generation Computer Systems**, v. 53, p. 25–42, 2015. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X15001995>. Citado na página 16.

SINGH, D.; REDDY, C. A survey on platforms for big data analytics. **Journal of Big Data**, v. 2, 12 2014. Citado na página 17.

SPARK, A. **Spark Configuration**. 2022. Disponível em: <https://spark.apache.org/docs/3.3.0/configuration.html>. Citado 2 vezes nas páginas 21 e 22.

SPARK, A. **Spark overview**. 2022. Disponível em: <https://spark.apache.org/3.3.0/latest/>. Citado 2 vezes nas páginas 13 e 17.

TSAI, C. et al. Big data analytics: a survey. **Journal of Big Data**, Springer Open, v. 2, n. 1, dez. 2015. ISSN 2196-1115. Funding Information: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the paper. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST103-2221-E-197-034, MOST104-2221-E-197-005, and MOST104-2221-E-197-014. Publisher Copyright: © 2015, Tsai et al. Citado na página 17.

WANG, G.; XU, J.; HE, B. 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)**. [S.l.: s.n.], 2016. p. 586–593. Citado 4 vezes nas páginas 13, 14, 21 e 34.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3. ed. Amsterdam: Morgan Kaufmann, 2011. (Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-374856-0. Disponível em: <http://www.sciencedirect.com/science/book/9780123748560>. Citado na página 38.

ZAHARIA, M. et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: **Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation**. USA: USENIX Association, 2012. (NSDI'12), p. 2. Citado 2 vezes nas páginas 13 e 18.

ZAHARIA, M. et al. Spark: Cluster computing with working sets. In: **Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing**. USA: USENIX Association, 2010. (HotCloud'10), p. 10. Citado 2 vezes nas páginas 17 e 21.

ANEXO A – PLANO EXPERIMENTAL

# Experimento	X1	X2	X3	X4	X5	X6	X7
1	2000	2000	16000	8	96000	2	16
2	16	2000	1000000	200	96000	2	4000
3	16	16	1000000	8	24000	6	4000
4	2000	16	1000000	200	24000	6	4000
5	16	2000	16000	8	24000	2	16
6	2000	2000	1000000	8	24000	6	4000
7	16	16	1000000	200	96000	6	4000
8	2000	2000	1000000	200	24000	6	16
9	16	2000	1000000	8	96000	6	4000
10	16	16	16000	200	96000	2	4000
11	2000	2000	1000000	8	96000	6	16
12	2000	16	16000	200	24000	2	4000
13	16	16	1000000	200	24000	2	4000
14	2000	2000	16000	200	96000	2	4000
15	2000	16	16000	8	96000	6	16
16	16	2000	1000000	200	96000	6	16
17	2000	2000	16000	8	96000	6	4000
18	16	16	1000000	200	24000	6	16
19	2000	2000	16000	200	24000	6	4000
20	2000	16	1000000	8	24000	2	4000
21	16	2000	16000	200	96000	2	16
22	16	2000	16000	8	24000	6	4000
23	2000	2000	16000	200	96000	6	16
24	16	16	1000000	8	24000	2	16
25	16	16	16000	8	96000	6	4000
26	16	2000	16000	8	96000	2	4000
27	2000	16	1000000	8	24000	6	16
28	16	2000	1000000	8	24000	6	16
29	16	16	16000	200	96000	6	16
30	16	16	1000000	8	96000	6	16
31	2000	16	1000000	200	96000	6	16
32	2000	2000	16000	8	24000	2	4000
33	16	2000	16000	200	24000	6	16
34	16	2000	16000	8	96000	6	16
35	2000	16	1000000	200	24000	2	16
36	16	16	1000000	8	96000	2	4000
37	2000	2000	1000000	8	24000	2	16
38	16	2000	1000000	8	96000	2	16
39	2000	16	16000	8	24000	2	16
40	16	2000	1000000	200	24000	2	16
41	16	16	16000	200	24000	2	16
42	2000	16	1000000	8	96000	6	4000
43	16	2000	1000000	8	24000	2	4000
44	16	16	16000	8	96000	2	16
45	2000	2000	1000000	200	96000	6	4000
46	16	2000	16000	200	96000	6	4000
47	16	16	1000000	200	96000	2	16
48	16	2000	1000000	200	24000	6	4000
49	2000	16	16000	8	24000	6	4000
50	2000	16	16000	200	24000	6	16
51	2000	16	1000000	200	96000	2	4000
52	16	16	16000	8	24000	6	16
53	16	16	16000	8	24000	2	4000
54	2000	16	1000000	8	96000	2	16
55	2000	2000	1000000	200	24000	2	4000
56	2000	2000	16000	200	24000	2	16

57	2000	2000	16000	8	24000	6	16
58	16	16	16000	200	24000	6	4000
59	16	2000	16000	200	24000	2	4000
60	2000	2000	1000000	200	96000	2	16
61	2000	2000	1000000	8	96000	2	4000
62	2000	16	16000	200	96000	6	4000
63	2000	16	16000	200	96000	2	16
64	2000	16	16000	8	96000	2	4000
65	16	2000	1000000	200	24000	6	4000
66	2000	2000	1000000	8	96000	6	16
67	2000	16	16000	200	96000	6	4000
68	16	2000	16000	200	96000	2	16
69	16	2000	16000	8	24000	6	4000
70	16	2000	16000	200	24000	6	16
71	16	16	16000	8	96000	2	16
72	16	16	16000	200	96000	6	16
73	2000	2000	16000	200	24000	2	16
74	16	2000	16000	8	24000	2	16
75	2000	2000	16000	200	24000	6	4000
76	16	2000	1000000	200	96000	6	16
77	16	16	1000000	8	24000	2	16
78	2000	2000	16000	200	96000	2	4000
79	16	2000	16000	8	96000	2	4000
80	2000	16	16000	200	24000	6	16
81	2000	2000	16000	8	96000	6	4000
82	2000	2000	1000000	200	96000	2	16
83	16	16	16000	200	24000	6	4000
84	2000	2000	1000000	200	24000	6	16
85	16	2000	1000000	200	24000	2	16
86	2000	16	1000000	200	24000	2	16
87	16	16	1000000	200	96000	6	4000
88	16	16	1000000	200	96000	2	16
89	2000	2000	1000000	200	96000	6	4000
90	16	2000	16000	8	96000	6	16
91	2000	2000	1000000	8	96000	2	4000
92	16	2000	1000000	8	24000	6	16
93	16	16	16000	8	24000	6	16
94	16	2000	1000000	200	96000	2	4000
95	16	16	1000000	200	24000	6	16
96	16	16	16000	8	96000	6	4000
97	2000	16	16000	8	24000	6	4000
98	2000	16	1000000	200	24000	6	4000
99	2000	2000	1000000	8	24000	2	16
100	16	2000	1000000	8	24000	2	4000
101	2000	16	16000	8	96000	2	4000
102	2000	2000	16000	8	96000	2	16
103	16	16	16000	8	24000	2	4000
104	16	16	1000000	200	24000	2	4000
105	16	16	1000000	8	96000	2	4000
106	2000	16	1000000	8	96000	6	4000
107	2000	2000	1000000	200	24000	2	4000
108	16	16	16000	200	24000	2	16
109	2000	2000	16000	8	24000	6	16
110	2000	16	16000	200	96000	2	16
111	2000	16	1000000	200	96000	6	16
112	2000	16	1000000	8	24000	6	16
113	2000	16	1000000	8	24000	2	4000

114	2000	2000	16000	200	96000	6	16
115	16	16	1000000	8	96000	6	16
116	16	2000	1000000	8	96000	6	4000
117	2000	2000	16000	8	24000	2	4000
118	16	2000	16000	200	24000	2	4000
119	16	2000	1000000	8	96000	2	16
120	2000	2000	1000000	8	24000	6	4000
121	2000	16	16000	8	96000	6	16
122	2000	16	16000	200	24000	2	4000
123	2000	16	1000000	8	96000	2	16
124	16	2000	16000	200	96000	6	4000
125	16	16	16000	200	96000	2	4000
126	2000	16	16000	8	24000	2	16
127	2000	16	1000000	200	96000	2	4000
128	16	16	1000000	8	24000	6	4000
129	16	2000	16000	8	96000	2	4000
130	16	16	1000000	8	24000	2	16
131	16	2000	1000000	8	24000	6	16
132	2000	16	16000	200	96000	2	16
133	2000	2000	16000	200	96000	2	4000
134	2000	2000	16000	200	96000	6	16
135	16	2000	16000	200	96000	6	4000
136	2000	2000	1000000	200	96000	2	16
137	2000	2000	16000	200	24000	6	4000
138	16	16	1000000	8	24000	6	4000
139	2000	16	16000	8	24000	2	16
140	16	16	1000000	8	96000	2	4000
141	2000	16	16000	200	24000	6	16
142	16	2000	16000	8	96000	6	16
143	2000	2000	16000	8	24000	2	4000
144	16	2000	16000	200	96000	2	16
145	2000	16	1000000	8	96000	2	16
146	16	16	16000	8	96000	2	16
147	16	16	16000	8	24000	6	16
148	16	2000	16000	8	24000	2	16
149	2000	16	1000000	200	96000	6	16
150	2000	16	1000000	200	24000	6	4000
151	16	16	16000	8	96000	6	4000
152	2000	2000	16000	200	24000	2	16
153	2000	16	16000	200	96000	6	4000
154	2000	16	16000	200	24000	2	4000
155	16	16	16000	200	96000	2	4000
156	16	2000	1000000	200	24000	2	16
157	16	2000	1000000	8	24000	2	4000
158	2000	16	1000000	200	24000	2	16
159	16	16	16000	8	24000	2	4000
160	16	16	1000000	200	96000	6	4000
161	16	16	1000000	200	24000	6	16
162	16	16	16000	200	24000	6	4000
163	2000	16	16000	8	96000	2	4000
164	2000	16	1000000	200	96000	2	4000
165	2000	2000	1000000	200	24000	2	4000
166	16	16	16000	200	24000	2	16
167	16	2000	1000000	200	96000	6	16
168	16	2000	16000	200	24000	6	16
169	2000	2000	16000	8	24000	6	16
170	16	2000	1000000	8	96000	6	4000

171	16	16	1000000	200	96000	2	16
172	2000	2000	1000000	8	96000	2	4000
173	2000	2000	16000	8	96000	6	4000
174	2000	16	16000	8	96000	6	16
175	2000	16	1000000	8	24000	6	16
176	2000	2000	1000000	200	24000	6	16
177	16	2000	16000	200	24000	2	4000
178	2000	2000	1000000	8	96000	6	16
179	16	2000	1000000	200	96000	2	4000
180	16	16	1000000	200	24000	2	4000
181	16	16	1000000	8	96000	6	16
182	16	16	16000	200	96000	6	16
183	16	2000	16000	8	24000	6	4000
184	16	2000	1000000	8	96000	2	16
185	2000	2000	1000000	8	24000	2	16
186	2000	2000	16000	8	96000	2	16
187	2000	16	1000000	8	96000	6	4000
188	2000	2000	1000000	200	96000	6	4000
189	16	2000	1000000	200	24000	6	4000
190	2000	2000	1000000	8	24000	6	4000
191	2000	16	16000	8	24000	6	4000
192	2000	16	1000000	8	24000	2	4000
193							

LEGENDA:
X1: spark.shuffle.file.buffer
X2: spark.io.compression.lz4.blockSize
X3: spark.sql.files.maxPartitionBytes
X4: spark.sql.shuffle.partitions
X5: spark.reducer.maxSizeInFlight
X6: spark.default.parallelism
X7: spark.broadcast.blockSize