Habilitando previsões de desastres em cenários de IoT com transferência de aprendizado

Marcus Artiaga Colantoni

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2022

Marcus Artiaga Colantoni

Habilitando previsões de desastres em cenários de IoT com transferência de aprendizado

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Rafael Pasquini
Coorientador: Prof. Dr. Rodrigo Sanches Miani

Uberlândia
2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

C683 Colantoni, Marcus Artiaga, 1989-
2022 Habilitando previsões de desastres em cenários de IoT com transferência de aprendizado [recurso eletrônico] / Marcus Artiaga Colantoni. - 2022.

Orientador: Rafael Pasquini.
Coorientador: Rodrigo Sanches Miani.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.617
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Pasquini, Rafael,1981-, (Orient.). II. Miani, Rodrigo Sanches,1983-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado 21/2022, PPGCO				
Data:	31 de outubro de 2022	Hora de início:	14:00	Hora de encerramento:	16:30
Matrícula do Discente:	11912CCP021				
Nome do Discente:	Marcus Artiaga Colantoni				
Título do Trabalho:	Habilitando previsões de desastres em cenários e IoT com transferência de aprendizado				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Prof. Dr. Bruno Bogaz Zarpelão - UEL; Prof. Dr. Paulo Rodolfo da Silva Leite Coelho - FACOM/UFU; Prof. Dr. Rodrigo Sanches Miani - FACOM/UFU e Prof. Dr. Rafael Pasquini - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Bruno Bogaz Zarpelão - Londrina/PR; Paulo Rodolfo da Silva Leite Coelho, Rodrigo Sanches Miani e Rafael Pasquini - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Rafael Pasquini, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Paulo Rodolfo da Silva Leite Coelho**, **Professor(a) do Magistério Superior**, em 16/11/2022, às 14:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 16/11/2022, às 15:52, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Pasquini**, **Professor(a) do Magistério Superior**, em 16/11/2022, às 18:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Bruno Bogaz Zarpelão**, **Usuário Externo**, em 16/11/2022, às 23:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4070566** e o código CRC **42EE28EA**.

Agradecimentos

Agradeço a Deus, autor da minha vida, a quem devo tudo o que tenho e o que sou.

Agradeço à minha esposa, Jeanne Gonçalves Colantoni, por todo o amor, apoio, companheirismo e compreensão. E pela sua companhia dia a dia que torna minha vida melhor e me ajuda a superar desafios. Seu apoio e carinho demonstrados a todo tempo foram fundamentais para o sucesso deste trabalho.

Agradeço ao meu filho Kaleb, que mesmo sendo uma criança me instiga a continuar evoluindo como profissional e acadêmico.

Agradeço aos meus pais, por serem exemplos de caráter e humanidade. O amor, o cuidado e o sacrifício empenhados no meu ensino e de minha irmã, nos deram valores e educação que o mundo jamais poderá tirar. A vocês eu devo toda a minha formação.

Agradeço ao professor Rafael Pasquini, meu orientador, e ao professor Rodrigo Sanches Miani, meu coorientador. O apoio e a dedicação exemplar demonstrados na minha orientação aumentam ainda mais o respeito e a admiração que tenho pelo trabalho de vocês.

"O importante é não parar de questionar."
(Albert Einstein)

Resumo

A tecnologia da informação e da comunicação tem mudado a forma como a humanidade vê o mundo ao seu redor. Diante da maturação da Internet das Coisas (IoT) e das possibilidades ofertadas pela inteligência artificial, temos maior capacidade de observar e de estimar, coisas e situações, a partir de informações coletadas no ambiente. Este trabalho investiga o uso de transferência de aprendizado como ferramenta habilitadora para o monitoramento de locais remotos, visando à prevenção de desastres. O objetivo principal é prover uma solução viável para cenários com restrição de largura de banda para comunicação de dados, como zonas rurais por exemplo. Como forma de validar a arquitetura proposta, apresentamos resultados obtidos em um protótipo construído em nossos laboratórios que permite avaliar a acurácia de modelos de aprendizado de máquina em diferentes condições de transmissão de dados de monitoramento. Os resultados demonstram a eficácia da técnica de transferência de aprendizado como um dos pilares para a escalabilidade de um sistema de monitoramento de desastres.

Palavras-chave: Aprendizado de máquina, IoT, Transferência de aprendizado, Gerenciamento de desastres, infraestrutura virtualizada.

Abstract

Information and communication technology have changed the way humanity sees the world around it. Faced with the maturation of the Internet of Things (IoT) and the possibilities offered by artificial intelligence, we have a greater ability to observe and estimate things and situations, based on information collected in the environment. This work investigates the use of transfer of learning as an enabling tool for monitoring remote locations, aiming at disaster prevention. The main objective is to provide a viable solution for scenarios with limited bandwidth for data communication, such as rural areas. As a way to validate the proposed architecture, we present results obtained in a prototype built in our laboratories that allows us to evaluate the accuracy of machine learning models under different conditions for the transmission of monitoring data. The results demonstrate the effectiveness of the transfer of learning technique as one of the pillars for the scalability of a disaster monitoring system.

Keywords: Machine learning, IoT, transfer of learning, disasters.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ADMITS *Architecting Distributed Monitoring and Analytics for IoT in Disaster Scenarios*

API *Application Programming Interface*

AM *Machine Learning*

CNPq *Conselho Nacional de Desenvolvimento Científico e Tecnológico*

DB *Data Base*

DL *Deep Learning*

DS *Data Science*

DSL *Digital Subscriber Line*

DDoS *Distributed denial of service*

FACTI *Fundação de Apoio à Capacitação em Tecnologia da Informação*

FL *Federated Learning*

GUI *Graphical User Interface*

HTTP *HyperText Transfer Protocol*

IA *Inteligência Artificial*

IBM *International Business Machines Corporation*

ICMP *Internet Control Message Protocol*

IOT *Internet Of Things*

IP *Internet Protocol*

JSON *JavaScript Object Notation*

LWM2M *Lightweight Machine to Machine*

LAN *Local Area Network*

M2M *Machine to Machine*

MQTT *Message Queuing Telemetry Transport*

NMAE *Normal Mean Absolute Error*

QoS *Quality Of Service*

RF *Random Forest*

RFG *Random Forest Regression*

SCADA *Supervisory Control and Data Acquisition*

TC *Traffic Control*

TCP *Transmission Control Protocol*

TTL *Time To Live*

TIC *Tecnologias da Informação e Comunicação*

UDP *User Datagram Protocol*

VM *Virtual Machine*

WAN *Wide Area Network*

Sumário

Capítulo 1

Introdução

Atualmente a tecnologia da informação e da comunicação (TIC) é responsável por gerenciar, manter e disponibilizar diversos serviços utilizados por toda a humanidade. No momento em que esta dissertação está sendo redigida, ocorre a maturação de um novo paradigma, chamado de Internet das Coisas, mais conhecido por seu acrônimo em inglês - *IoT* (*Internet of Things*).

A cada ano, milhões de novos dispositivos são inseridos na rede mundial de computadores, sendo cada vez mais comum que equipamentos que antes possuíam funções simples com mecanismos analógicos, passem a ter a denominação de *smart*, contando com a adição de alguma funcionalidade que se beneficie da Internet (CANAVILHAS, 2009).

Naturalmente, dado o aumento significativo e cada vez mais acelerado na quantidade de dispositivos ingressando na Internet, a infraestrutura se sobrecarrega de maneira proporcional e surgem diversos desafios tecnológicos, por exemplo, quanto a largura de banda nos enlaces de dados e a interoperabilidade de tecnologias, frente a fluxos massivos de dados(JUNIOR[1]; MORENO[1], 2015).

Neste trabalho consideramos o uso de IoT como ferramenta essencial para o monitoramento de ambientes complexos, ou seja, ambientes que requerem minuciosa observação de diversos parâmetros, coletados a partir de inúmeros sensores. Através deste minucioso monitoramento, objetiva-se ofertar sistemas de monitoramento que permitam detectar desastres, sejam eles naturais ou causados pelo homem.

No entanto, a análise deste conjunto formado por um elevado número de amostras coletadas a partir de inúmeros sensores apresenta alguns desafios importantes. Primeiramente, a coleta e o escoamento de todas as amostras monitoradas pode ser inviabilizada em algumas localidades, cuja qualidade da infraestrutura (transmissão de dados, energia elétrica, etc) disponível seja deficitária. Em segundo lugar, a interpretação de todos estes dados não é trivial e requer um corpo de especialistas, capaz de diferenciar situações normais de situações adversas, ou seja, que indicam desastres. Eventualmente, este corpo de especialistas não está disponível a todo instante, para os diversos sistemas observados, por exemplo, em uma grande empresa ou país.

Neste contexto, uma proposta de arquitetura é apresentada nesta dissertação, utilizando técnicas de aprendizado de máquina como forma de habilitar a construção de ferramentas computacionais, não somente escaláveis, como também capazes de identificar desastres naturais a partir de modelos treinados com o auxílio de especialistas. A arquitetura proposta visa tornar o monitoramento, mesmo que a partir de zonas rurais com restrições de infraestrutura, escalável, na medida em que apenas um pequeno conjunto de dados precisa ser movimentado para a detecção de desastres. Ao mesmo tempo, o treinamento de modelos de aprendizado de máquina, a partir das amostras coletadas, visa automatizar a detecção de desastres, não demandando a presença constante de especialistas.

Segundo Pan et al. (2010), a Transferência de Aprendizado pode se configurar quando as tarefas de origem e os rótulos aos quais se deseja prever, são os mesmos, porém os domínios onde o modelo é aplicado são diferentes. Nesse sentido, Transferência de Aprendizado está relacionada a transferir o aprendizado gerado em um local de modo a reutiliza-lo em outro (PAN et al., 2010). Baseado nesta definição o presente trabalho irá aplicar conceitos da transferência de aprendizado como o pilar central da arquitetura proposta, visando habilitar o efetivo monitoramento e detecção de desastres.

Esta dissertação foi desenvolvida no contexto do projeto *ADMITS* (*Architecting Distributed Monitoring and Analytics for IoT in Disaster Scenarios*) "Arquitetura de monitoramento distribuído e análise para *IoT* em cenários de desastre", aprovada na chamada CAPES/STIC-AMSUD 2019 e integra colaboradores do Brasil, França, Chile e Uruguai (BRASIL, 2022). O *ADMITS* tem como um de seus objetivos a implementação de algoritmos, de protocolos e de arquiteturas que permitam que ambientes de computação distribuídos suportem monitoramento, detecção de falhas e análise em cenários de desastres. Dessa forma, a presente dissertação visa contribuir para o estado da arte do projeto *ADMITS*, bem como a criação de conteúdos técnicos para a comunidade científica.

1.1 Motivação

Segundo Gallizzi (2022), pelo mundo todo, ocorrem mudanças climáticas que elevam a intensidade de desastres naturais, como enchentes, tempestades e incêndios florestais. Buscando entender quais países estão mais suscetíveis a esses perigos, especialistas de energia da *Uswitch* (empresa britânica de energia) analisaram dados entre 1902 e 2021, que incluíam mais de 15 mil desastres naturais registrados. Uma pontuação total foi calculada, usando a classificação percentual média em quatro categorias diferentes: número de pessoas afetadas, fatalidades, desastres e danos. A Tabela 1 sumariza esses resultados (GALLIZZI, 2022).

O Brasil é o pais que completa a lista dos 10 dos países que mais sofrem desastres naturais, com pontuação de 9,217. Dois exemplos recentes de desastres naturais ocorridos

Tabela 1 – Top 10 mundial - desastres naturais. Fonte: Adaptado de (GALLIZZI, 2022)

Rank	País	Afetados	Fatalidades	Total Desastres	Prejuízo (US$)	Score/10
1	China	3,3 Bilhões	12,5 Milhões	982	799 Bilhões	9,978
2	Índia	2,5 Bilhões	9,1 Milhões	757	187 Bilhões	9,912
3	Bangladesh	464 Milhões	3 Milhões	356	39 Bilhões	9,720
4	Estados Unidos	116 Milhões	44 Mil	1090	17 Trilhões	9,648
5	Filipinas	239 Milhões	72 Mil	671	43 Bilhões	9,587
6	Indonésia	37 Milhões	243 Mil	567	47 Bilhões	9,543
7	Iran	58 Milhões	163 Mil	253	51 Bilhões	9,450
8	Paquistão	99 Milhões	179 Mil	234	42 Bilhões	9,442
9	Japão	23 Milhões	239 Mil	375	735 Bilhões	9,362
10	Brasil	116 Milhões	13 Mil	251	47 Bilhões	9,217

no Brasil envolvem o rompimento de barragens em Mariana (MG) em 2015 e Brumadinho (MG) em 2019.

De acordo com Martinez (2022), em Mariana, a catástrofe ocasionou 19 mortes e, em Brumadinho, a perspectiva é de aproximadamente 270 mortos e 11 pessoas ainda desaparecidas. A avaliação do rompimento das barragens é dividida em dois momentos, o primeiro com grande impacto sobre as populações, as pessoas que morreram; e o segundo impacto, à enorme quantidade de resíduos lançados no meio ambiente. Os impactos sobre a biota são complicados para serem medidos, já que estima-se que nos próximos cinquenta anos ainda serão observados as consequências desse desastre.

No caso das barragens as fiscalizações são realizadas por um fiscal, que realiza a aferição a intervalos pré-determinados. Não existe um sistema que reúna os dados ou concentre as informações de modo a estimar ou a prever situações criticas das barragens. Essa mesma visão pode ser aplicada a diversas outras áreas no Brasil, tanto no tocante a desastres que são ocasionados por intervenção humana como para casos de incidentes causados pela natureza. O Brasil ainda sofre com várias regiões em que a infraestrutura de TIC é precária ou até mesmo inexistente, com constantes oscilações nos enlaces de dados, o que dificulta o tráfego massivo de informações. Essas barreiras acabam inviabilizando a implantação de um sistema efetivo de monitoramento para mitigar desastres.

1.2 Objetivos e Desafios da Pesquisa

Esse trabalho tem como principal objetivo prover uma solução viável de monitoramento a cenários de desastre, investigando a técnica de transferência de aprendizado como ferramenta habilitadora dessa solução.

A construção de um protótipo virtualizado para realização de experimentos e a implementação de algoritmos de inteligência artificial para manipular os dados coletados compõem os objetivos técnicos.

A construção do protótipo virtualizado possibilita a realização dos experimentos necessários para provar a hipótese. A implementação de algoritmos de aprendizado de máquina em conjunto com a transferência de aprendizado serão o arcabouço deste trabalho, pois,

por meio deles, foi realizada uma analise empírica no tocante à viabilidade do monitoramento em regiões remotas e afastadas, onde a redução no volume de bytes e no número de pacotes trafegados na rede são fatores determinantes.

Os objetivos específicos deste trabalho são:

- ❏ Análise empírica da transferência de aprendizado como ferramenta habilitadora para monitoramento de locais remotos;
- ❏ Construção de uma arquitetura para ambiente *IoT* voltada para monitoramento de desastres com capacidade de emular distâncias geográficas e restrições de comunicação de dados;
- ❏ Implementação de algoritmos de aprendizado de máquina capazes de realizar previsões em cenários de desastres.

O cumprimento dos objetivos listados é desenvolvido nos próximos Capítulos da dissertação, conforme organização que está descrita na Seção 1.4.

1.3 Hipótese

A principal hipótese desta dissertação é a de que a combinação das tecnologias *IoT* e as técnicas de AM em conjunto com a transferência de aprendizado, geram sistemas com desempenho equivalente ao modelo padrão (descrito na Seção 4) que, mesmo em momentos de falha na infraestrutura, mantêm previsões de desastres.

O êxito nas etapas de transmissão de porções dos dados mantendo baixa dispersão na taxa de erro *NMAE*, que é uma medida utilizada para avaliar modelos de regressão, é o que garante o sucesso da proposta. No Capítulo 5 são apresentados os resultados dos testes dos dados trafegados pela infraestrutura construída, e a análise empírica entre os cenários de alta disponibilidade de recursos e os cenários de desastre.

1.4 Organização da Dissertação

A organização dos Capítulos desta dissertação são respectivamente: Capítulo 2 que descreve as tecnologias envolvidas na pesquisa, onde também é apresentado o estado da arte atual para estas. O Capítulo 3 são apresentados os trabalhos relacionados a essa dissertação. O Capítulo 4 apresenta a arquitetura proposta de forma específica, detalhando os componentes utilizados e seus aspectos de empregabilidade no projeto. No Capítulo 5 são descritos os experimentos realizados para verificar a eficácia da proposta, bem como as analises dos resultados gerados nesses experimentos. Por fim, no Capítulo 6, são apresentadas as conclusões e contribuições do trabalho para o estado da arte, as limitações da pesquisa realizada e as inspirações para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Revisão do *ADMITS*

O projeto *ADMITS* (*Architecting Distributed Monitoring and Analytics for IoT in Disaster Scenarios*) "Arquitetura de monitoramento distribuído e análise para *IoT* em cenários de desastre", visa desenvolver algoritmos, protocolos e arquiteturas que permitam que ambientes de computação distribuídos suportem monitoramento, detecção de falhas e análise de cenários de desastres de *IoT*. Esse projeto tem implementações e incentivos pelo mundo todo, como se pode verificar em alguns cenários da América do Sul (PASQUINI et al., 2020b).

Anualmente, milhões de pessoas são afetadas por desastres naturais, causados pelo homem, e governos de todo o mundo investem recursos significativos em preparação, resposta imediata e reconstrução. O clima severo, as chuvas fortes, as inundações repentinas e os deslizamentos de terra afetam severamente a América do Sul desde novembro de 2015, deixando milhares de desabrigados e de mortos. No Brasil, o Serviço de Gerenciamento de Emergências fornece informações antecipadas a milhares de famílias afetadas por chuvas, deslizamentos de terra, seca.

Segundo o Sistema Nacional de Emergência (*SINAE*) do Uruguai, milhares de pessoas foram deslocadas pelas inundações causadas pelas fortes chuvas, além de tornados, que destruíram várias casas em diversas áreas do país. O Chile recebe atenção especial por ser o país mais propenso a desastres naturais. Quanto a terremotos, o Chile é um dos países mais propensos no mundo, principalmente devido à sua localização ao longo do Anel de Fogo do Pacífico. Situado em uma área de intensa atividade vulcânica e de terremotos, o país também é afetado por secas, inundações, tsunamis, erupções vulcânicas, incêndios florestais e incêndios.

Na gestão eficiente de cenários de desastre, o paradigma da *IoT* tem sido amplamente utilizado, como vemos em (SACCO et al., 2020). Em casos de desastre, a infraestrutura de comunicação de dados pode apresentar falhas, o que pode ocasionar perda de informações. É esperado que, nos próximos anos, o volume de dados gerados na Internet seja maior

do que é hoje, tornando o processamento e a análise desses dados mais complexos e com maior custo computacional.

A proposta do *ADMITS* é baseada na experiência das equipes da França, do Brasil, do Uruguai e do Chile em vários campos complementares de pesquisa. Esses campos incluem computação e algoritmos distribuídos em larga escala, tolerância a falhas, sistemas dinâmicos e heterogêneos, processamento e análise de dados em tempo real, gerenciamento de dados em larga escala, sistemas auto-organizados, redes de computadores e coleta de informações de geolocalização pós-catástrofe.

2.2 Aplicação da *IoT* em Cenários de Desastres

Segundo Rosa (2020) Esta nova abordagem tecnológica, mais conhecida como *IoT*, utiliza recursos de Tecnologia da Informação e de Comunicação TIC em conjunto com infraestrutura de telecomunicação que provê os recursos de Internet. Existem projetos que podem ser implementados com hardware simplificado de baixo de custo, viabilizando assim trabalhos nesta linha. Também existem orçamentos milionários investidos em projetos *IoT*, devido a vasta área de aplicação e implementação desta abordagem (ROSA, 2020).

A *IoT* pode ser descrita como uma rede de objetos físicos dotados de tecnologia embarcada, como sensores conectados com a rede, capaz de reunir e de transmitir dados (PEREZ; PEREIRA, 2019). Pode ser vista como uma extensão da Internet atual, que possibilita dispositivos, utilizados no dia a dia, por mais diversos que possam ser, mas que tenham capacidade computacional e de comunicação de se conectarem na Internet. Essa conectividade possibilita comunicação entre estes dispositivos, de modo que sejam utilizados como provedores de serviços.

Novas funções agregadas a esses objetos comuns abrem caminho a inúmeras possibilidades, tanto no âmbito acadêmico, comercial quanto no industrial, unindo-se a equipamentos eletrônicos e eletromecânicos de automação industrial. Todavia, acompanhados de um número extenso de possibilidades, existem riscos e implicações, que criam grandes desafios sociais, tanto na definição de políticas regulamentárias como nos aspectos técnicos.

As aplicações da *IoT* podem melhorar a qualidade de vida das pessoas, especialmente no que diz respeito à preservação de vidas em cenários de desastres. Por exemplo, um sistema de monitoramento regional poderia ser utilizado para geração de alertas à comunidade, quando os dispositivos de sensoriamento indicarem mudanças que sugerem início ou grande probabilidade da ocorrência de um desastre (PEREZ; PEREIRA, 2019).

A *IoT* torna possível fazer o monitoramento de diversos tipos de cenários por meio da coleta de dados de sensores, de dispositivos e de microchips de forma autônoma. A vantagem na utilização de sistemas contínuos é que eles podem operar sem a interação direta de um operador humano, permitindo maior operabilidade e fluxo contínuo de operação

do sistema (PEREZ; PEREIRA, 2019).

Para um maior aproveitamento das possibilidades proporcionadas pelo *IoT*, existem pré-requisitos tecnológicos a serem seguidos, como infraestruturas seguras e confiáveis, pois a integridade das informações se torna um fator crítico na tomada de decisão em fluxo de monitoramento. O *IoT* também tem por padrão a geração de um grande tráfego de rede, isso se deve ao alto número de dispositivos e de sensores que são englobados nas soluções de *IoT* (AHMED et al., 2016).

Na ótica do projeto *ADMITS*,procura-se o desenvolvimento de uma arquitetura na qual as tecnologias de *IoT*, computação de borda e computação de névoa participem para fornecer os recursos necessários para análise de dados e monitoramento de falhas, para ambientes potencialmente sujeitos a desastres (STADLER; PASQUINI; FODOR, 2017). Nos casos das tragédias, ocorridas nas cidades de Mariana e Brumadinho em Minas Gerais, decorrentes do rompimento das barragens, muitas pessoas morreram e tiveram suas casas e trabalho destruídos, além de as áreas afetadas se tornaram inabitáveis (VANNUCHI, 2022).

Esses dois desastres servem de inspiração na busca por soluções de monitoramento que possam apresentar previsões com certa antecedência, utilizando os recursos de *IoT* como, por exemplo, a emissão de um alerta aos moradores da região, podendo assim salvar, muitas vidas. O projeto *ADMITS*, juntamente com as tecnologias ofertadas por *IoT*, busca desenvolver soluções capazes de mitigar os cenários de desastre.

2.3 Tecnologias

Nesta seção serão apresentadas as principais tecnologias que constituem a construção da infraestrutura virtualizada, como a instância do *dojot*; o *Mosquitto*, que é o software de código aberto, que irá realizar o transporte das informações por meio do protocolo *MQTT* para o agente *IoT* e o *docker*, que é o orquestrador de contêineres *LINUX*, que foi utilizado como base para a execução de todas as soluções que compõem o *dojot*.

2.3.1 *Dojot*

O *dojot* nasceu com o objetivo de desenvolver e de demonstrar tecnologias para as cidades inteligentes. Os desenvolvedores têm como foco os pilares de segurança pública, a mobilidade urbana e de saúde e pretendem construir um ecossistema multidisciplinar nessas áreas. O *dojot* é uma plataforma brasileira com uma proposta de software aberto para facilitar o desenvolvimento de soluções utilizando *IoT* com conteúdo local, voltado às necessidades brasileiras, assumindo um papel habilitador. O *dojot* possui (DOJOT, 2022):

- *APIs* abertas para acesso às aplicações e aos recursos da plataforma;

- ❏ Gerenciamento do ciclo de vida de dispositivos (planejamento, configuração e monitoramento);
- ❏ Construção de fluxos de dados e de regras de forma visual para processamento de dados em tempo real;
- ❏ Persistência dos dados;
- ❏ Interface para acesso aos dados em tempo real.

Baseado em tecnologias no estado da arte, o *dojot* é uma plataforma que adota uma arquitetura de micro serviços, possibilitando implantação funcional customizada e escalável conforme a necessidade. Essa plataforma possui integração com outros sistemas baseados em serviços e micro serviços. Apesar de o foco original da plataforma ser o ecossistema de cidades inteligentes, o *dojot* tem sido utilizado com sucesso no desenvolvimento de soluções de *IoT* nas áreas de saúde, no agronegócio e na indústria (DOJOT, 2022).

O *dojot* é resultado do projeto "Plataforma Aberta para *IoT* e suas Aplicações", que conta com o apoio do FUNTTEL, do MCTIC, via Finep e é conduzido pelo CPQD, em parceria com outras instituições de ciência e tecnologia como Instituto Atlântico, Centro de Tecnologia da Informação Renato Archer (CTI) e Fundação de Apoio à Capacitação em Tecnologia da Informação (FACTI). Com código aberto, está posicionado para ser habilitador e facilitador do desenvolvimento de soluções *IoT*. Além disso, a plataforma busca disponibilizar a toda a comunidade as contribuições geradas na evolução e no aperfeiçoamento do estado da arte (DOJOT, 2022).

A arquitetura atual que guia a implementação do *dojot*, detalhando os componentes que compõem a solução, assim como as suas funcionalidades, é demonstrada na Figura 1. A solução é composta de uma série de tecnologias consolidadas, que juntas compõem o *dojot* (DOJOT, 2022).

O *dojot* foi projetado para tornar possível uma construção rápida, fornecendo uma plataforma de uso simplificado, escalável e também robusta. Sua arquitetura interna faz uso de muitos componentes conhecidos de código aberto, todavia existem algumas partes que o compõem que foram implementadas exclusivamente pela equipe *dojot* (DOJOT, 2022).

A utilização do *dojot* por um usuário poderia, por exemplo, configurar dispositivos de *IoT* por meio da *GUI* ou diretamente usando as *APIs REST* fornecidas pelo *API Gateway*. Os fluxos de processamento de dados também podem ser configurados, e essas entidades podem executar uma variedade de ações, como gerar notificações quando um atributo de dispositivo específico atingir um determinado limiar. Essas *APIs* podem ainda, salvar todos os dados gerados por um dispositivo em um banco de dados externo, à medida que os dispositivos começam a enviar suas leituras para *dojot*, um usuário pode (DOJOT, 2022):

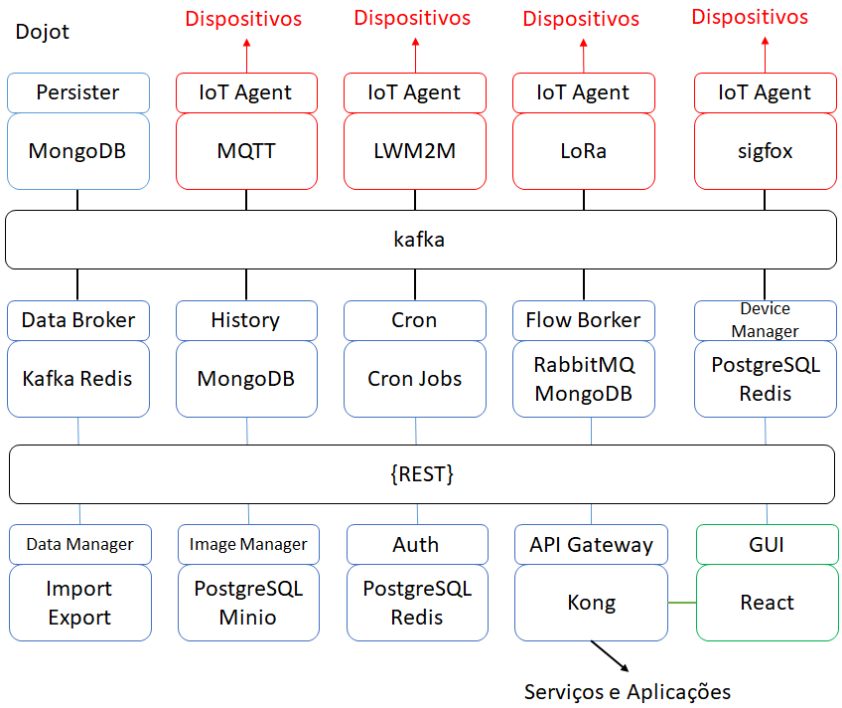

Figura 1 – A arquitetura de microsserviço da plataforma *dojot*. Fonte: Adaptado de (DOJOT, 2022).

- ❏ receber essas leituras em tempo real pelos canais *socket.io ou websocket*;
- ❏ consolidar todos os dados em dispositivos virtuais;
- ❏ reunir todos os dados do banco de dados, histórica e subsequentemente.

Por meio da *APIs REST*, esses recursos estão disponíveis e são os blocos de construção básicos que qualquer aplicativo, baseado no *dojot*, deve utilizar. A *GUI* fornece uma maneira fácil de executar operações de gerenciamento para todas as entidades relacionadas à plataforma como, por exemplo, usuários, dispositivos, modelos e fluxos, e também pode ser usada para verificar se tudo está funcionando corretamente (DOJOT, 2022).

Não há compartilhamento de dados e o contexto do usuário é isolado, as credenciais de acesso são validadas pelo serviço de autorização para cada operação (solicitação da *API*). Dessa forma, um usuário, pertencente a um contexto específico, não poderá acessar nenhum dado de outros usuários (DOJOT, 2022).

Após a configuração dos dispositivos, o *IoT Agent* é capaz de mapear os dados recebidos dos mesmos e encapsulá-los em mensagens *MQTT*. Essas mensagens são enviadas ao *broker* para distribuição interna. Dessa forma, os dados chegam ao serviço de persistência, para que possam ser registrados em um banco de dados (DOJOT, 2022).

2.3.1.1 *Dojot - Kafka* e *DataBroker*

A plataforma distribuída de menssageria, Apache *Kafka*, pode ser utilizada por aplicações que precisam transmitir dados ou consumir/produzir em canais de dados. Em comparação com outras soluções de mensagens de código aberto, o *Kafka* parece ser mais apropriado para cumprir os requisitos de arquitetura do *dojot*, pois ele possui isolamento de responsabilidades, simplicidade e opção de compartilhamento, estando de acordo com a filosofia de segurança abordada pelo *dojot*. No *Kafka*, utiliza-se uma estrutura de tópicos especializada para garantir isolamento de dados, de usuários e de aplicações, viabilizando uma arquitetura *multi-tenant* (DOJOT, 2022).

Visando eficiência, o serviço *DataBroker* utiliza um banco de dados na memória, que adiciona contexto ao Apache *Kafka*, possibilitando que serviços internos ou externos possam consumir dados em tempo real com base no contexto (DOJOT, 2022).

2.3.1.2 *DeviceManager*

O *DeviceManager* é uma entidade central responsável por manter as estruturas de dados de dispositivos e de modelos, além de ser responsável por publicar quaisquer atualizações para todos os componentes interessados por meio do Kafka. O serviço não mantém estados, e tem seus dados persistidos em banco de dados, onde suporta isolamento de dados por contextos e aplicações, viabilizando uma arquitetura de *middleware* com múltiplos contextos (DOJOT, 2022).

2.3.1.3 Agente *IoT*

O agente *IoT* no *dojot* é um serviço de adaptação entre dispositivos físicos, e os componentes principais do *dojot* podem ser entendidos como um *driver* de dispositivo para um conjunto de ativos de rede. A plataforma *dojot* pode ter vários agentes *IoT*, cada um deles especializado em um protocolo específico como, por exemplo, *MQTT/JSON, CoAP/LWM2M, Lora/ATC, Sigfox/WDN e HTTP/JSON*. A comunicação por meio de canais seguros com dispositivos também é responsabilidade dos agentes *IoT* (DOJOT, 2022).

2.3.1.4 Serviço de Autorização de Usuários

Serviço que implementa o gerenciamento de perfil de usuários e o controle de acesso. Basicamente qualquer chamada de aplicação por meio do *API Gateway* é validada por esse serviço. Para ser capaz de atender a um grande volume de chamadas de autorização, esse serviço faz uso de cache, não mantém estados e pode ser escalado horizontalmente. Seus dados são mantidos em banco de dados, que podem ser configurados em *cluster* (DOJOT, 2022).

2.3.1.5 *Flowbroker*

Esse serviço provê mecanismos para construir fluxos de processamento de dados e para execução de um conjunto de ações. Os fluxos podem ser estendidos, usando um bloco de processamento externo, que pode ser incluído, ao se utilizar *APIs REST* (DOJOT, 2022).

2.3.1.6 *Kafka2Ftp*

O serviço *kafka2ftp*, permite o encaminhamento de mensagens do Apache *Kafka*, para servidores *FTP*. Ele se inscreve no tópico *tenant.dojot.ftp* do *Kafka*, onde as mensagens devem seguir um esquema específico. As mensagens podem ser redirecionadas para esses tópicos, usando um nó específico do *flowbroker* (DOJOT, 2022).

2.3.1.7 *Persister/History*

O componente *Persister* funciona como um condutor de dados e de eventos que devem ser persistidos em um banco de dados. Esses dados são convertidos em uma estrutura de armazenamento que é enviada para o banco de dados correspondente. Para armazenamento interno, utiliza-se uma base de dados não relacional, *MongoDB*, que pode ser configurada em modo *Sharded Cluster*, dependendo do caso de uso. Os dados persistentes podem ser consultados por meio de uma *API Rest*, fornecida pelo micro serviço *"history"* (DOJOT, 2022).

2.3.1.8 InfluxDB Storer e Retriever

Os serviços *InfluxDB Storer* e *InfluxDB* Retriever trabalham juntos, o *InfluxDB Storer* é responsável por consumir dados *Kafka*, de dispositivos *dojot*, e gravá-los no *InfluxDB*. Já o *InfluxDB Retriever* tem a função de obter os dados que foram escritos pelo *InfluxDB Storer*, no *InfluxDB* via *API REST* (DOJOT, 2022).

2.3.1.9 *Kong API Gateway*

O *Kong API Gateway* é utilizado como um ponto de fronteira entre as aplicações e os serviços externos e os serviços internos do *dojot*. Isso resulta em inúmeras vantagens como, por exemplo, ponto único de acesso e facilidade na aplicação de regras sobre as chamadas de *APIs*, sobre a limitação de tráfego e sobre o controle de acesso (DOJOT, 2022).

2.3.1.10 *GUI*

A Interface Gráfica de Usuário (*GUI*) no *dojot* é uma aplicação *web*, que provê interfaces responsivas para gerenciamento da plataforma, incluindo funcionalidades como (DOJOT, 2022):

- ❏ **Gerenciamento de perfil de usuários:** permite definir perfis e quais *APIs* podem ou não ser acessadas pelo respectivo perfil.
- ❏ **Gerenciamento de modelos de dispositivos:** operações de criação, visualização, edição e remoção.
- ❏ **Gerenciamento de dispositivos:** operações de criação, visualização (dispositivo e dados em tempo real), edição e remoção.
- ❏ **Gerenciamento de fluxos de processamento:** permite operações de criação, visualização, edição e remoção de fluxos de processamento de dados.
- ❏ **Notificações:** visualiza as notificações do sistema em tempo real e histórico unificados (DOJOT, 2022).

2.3.1.11 *Image manager*

Esse componente é responsável pelo armazenamento, e recuperação de imagens de *firmware* dos dispositivos. Ele é utilizado pelo mecanismo de atualização de *firmware* (DOJOT, 2022).

2.3.1.12 *X.509 Identity Management*

Este componente é responsável por atribuir identidades a dispositivos, tais identidades são representadas na forma de certificados x.509. Ele se comporta semelhante a uma Autoridade Certificadora *(CA)*, em que é possível submeter um *CSR* e receber um certificado de volta. Tendo o certificado instalado no dispositivo, é possível se comunicar de forma segura com a plataforma *dojot*, pois os dados coletados pelo dispositivo são trafegados por um canal seguro, que é criptografado, possibilitando garantir a integridade dos dados (DOJOT, 2022).

2.3.1.13 *Kafka WS*

Este componente é responsável por obter dados do Apache *Kafka*, via uma conexão *Websocket*. Foi desenhado para permitir que usuários do *dojot* possam recuperar dados brutos e/ou processados, em tempo real de dispositivos do *dojot* (DOJOT, 2022).

2.3.1.14 Infraestrutura

Outros componentes utilizados no *dojot* (DOJOT, 2022):

- ❏ ***Postgres*:** banco de dados, é utilizado para persistir informações de vários componentes, como do gerenciador de dispositivos por exemplo.

- ***Redis*:** banco de dados em memória, usado como cache em vários componentes, como o serviço de orquestração, gerenciador de subscrição, agentes *IoT* e outros.
- ***RabbitMQ*:** *broker* de mensagens, usado no orquestrador de serviços, para implementar fluxos de ação relacionados, que devem ser aplicados a mensagens recebidas de componentes.
- ***MongoDB*:** solução de banco de dados amplamente utilizada, fácil de usar e não adiciona *overhead* considerável nos locais onde foi empregado na *dojot*.
- **Zookeeper:** responsável por manter sob controle serviços replicados em *cluster* (DOJOT, 2022).

2.3.1.15 Comunicação

A comunicação em uma instância *dojot* ocorre basicamente de duas maneiras, por meio de requisições *HTTP* e/ou por meio de mensagens *Kafka* (DOJOT, 2022):

- **Por meio de requisições *HTTP*:** No caso de um componente necessitar recuperar dados de outro, como um agente *IoT* que precisa da lista de dispositivos configurados, do gerenciador de dispositivos, ele pode enviar uma requisição *HTTP* para o componente apropriado (DOJOT, 2022).
- **Por meio de mensagens *Kafka*:** Se um componente precisa enviar novas informações sobre um recurso controlado por ele, como novos dispositivos criados no gerenciador de dispositivos, o componente pode publicar esses dados por meio do *Kafka*. Utilizando-se esse mecanismo, qualquer outro componente que esteja interessado em tal informação precisa apenas ouvir um tópico específico para recebê-la. É valido notar que esse mecanismo não faça quaisquer associações fixas entre componentes, por exemplo, o gerenciador de dispositivos não sabe quais componentes precisam de suas informações, e um agente *IoT* não necessita saber qual componente está enviando dados por meio de um tópico específico (DOJOT, 2022).

2.3.2 Docker

O *docker* Docker-Inc (2022) é uma plataforma de código aberto, que facilita a criação e a administração de ambientes isolados, através do uso de contêineres. Dessa forma, é possível criar, implantar, copiar e migrar de um ambiente para outro com maior flexibilidade. A ideia do *docker* é a utilização de apenas uma máquina, em vez de várias. E, nessa única máquina, poder executar várias aplicações sem que haja conflitos entre elas. Essa tecnologia possui o mesmo nome da empresa que a criou, *docker Inc*, essa tecnologia é desenvolvida com base no trabalho realizado pela comunidade do *docker*, a comunidade que trabalha gratuitamente, para melhorar essas tecnologias em benefícios de todos. De

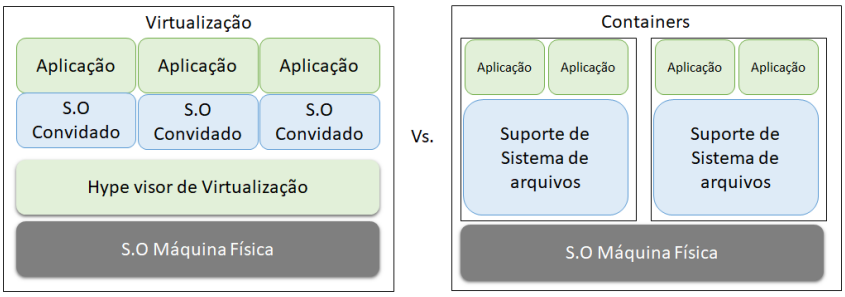

Figura 2 – MA Convencional vs *docker*. Fonte: Adaptado de (DOCKER-INC, 2022).

forma semelhante a uma máquina virtual, o *docker* é extremamente leve, mas não se trata de fato de uma máquina virtual, o *docker* utiliza contêineres que possuem uma arquitetura diferente, permitindo maior portabilidade e eficiência do que as máquinas virtuais convencionais. O container exclui a virtualização, e muda o processo para o *docker*, por isso não se pode dizer que o *docker* é uma máquina virtual. Na Figura 2, é possível visualizar a diferença arquitetônica entre o *docker* e uma virtualização.

Também é possível observar na Figura 2 que, na virtualização, temos um maior consumo de recursos, uma vez que para cada aplicação precisamos carregar um sistema operacional; já no *docker*, é possível verificar que não existe essa necessidade de múltiplos sistemas operacionais convidados.

Um contêiner é um ambiente isolado, contém um conjunto de processos que são executados a partir de uma imagem que fornece todos os arquivos necessários. Os contêineres compartilham o mesmo *Kernel* e isolam os processos da aplicação do restante do sistema. Com o *docker* é possível realizar configurações específicas em um ambiente isolado devido à facilidade para replicação de contêineres, o resultado é a criação de ambientes padronizados, tanto em desenvolvimento como em produção. Assim, é factível disponibilizar toda essa arquitetura para o cliente final, onde ele estiver, bastando replicar os contêineres, que serão executados da mesma maneira em qualquer lugar (DOCKER-INC, 2022).

Como o contêiner possui uma imagem, que contém todas as dependências de uma aplicação, ele é portátil e consistente em todas as etapas de desenvolvimento. Essa imagem é um modelo de somente leitura, que é utilizada para subir um contêiner. O *docker* também possibilita a criação de contêineres de imagens próprias, e a utilização delas como base para os contêineres (DOCKER-INC, 2022).

Mesmo tendo sido inicialmente desenvolvido com base na tecnologia *LXC* (*Linux* contêineres), o *docker* hoje é uma tecnologia independente de sistema operacional, logo pode ser utilizado em ambientes *Linux*, *Windows* e até mesmo *MacOS*(DOCKER-INC, 2022).

Como mostrado na Figura 3, o *docker* possui uma tecnologia de rede sobreposta, chamada rede *overlay*, que é uma rede que se sobrepõe à rede física, criando um sistema computacional distribuído, chamado de *Docker Swarm*. Dessa maneira, mesmo em um sistema distribuído, é possível que contêineres que estejam em diferentes máquinas ou geograficamente distribuídos pelo mundo, se comuniquem da mesma maneira que os de

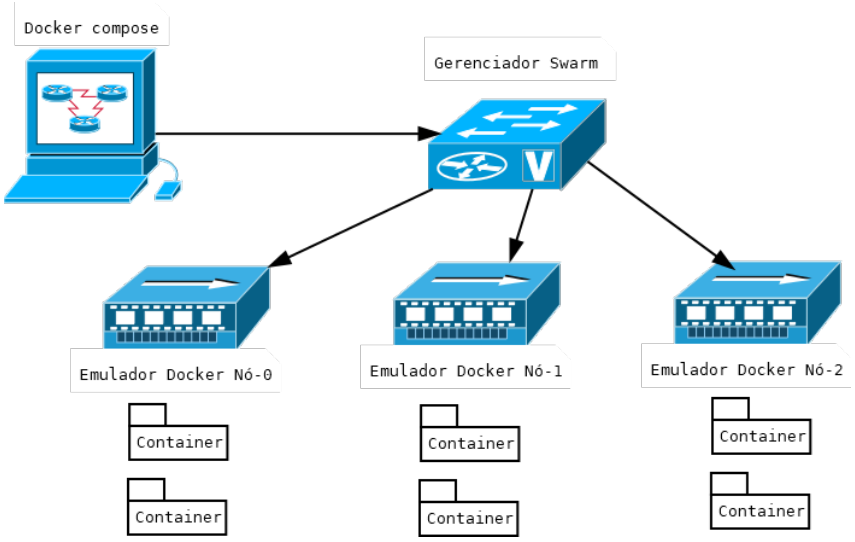

Figura 3 – Rede sobreposta *docker swarm*. Fonte: Adaptado de (DOCKER-INC, 2022).

uma rede local. O endereçamento *IP* pode utilizar uma faixa distinta da rede física, o que permite um escopo totalmente isolado. Essa abordagem gera alta escalabilidade para os contêineres *docker*, como também aumenta a segurança da comunicação entre eles (DOCKER-INC, 2022).

2.3.3 *Mosquitto*

O *Mosquitto* é um *Broker MQTT* (*Message Queue Telemetry Transport*), é *Open Source* e pode ser utilizado desde computadores de uso comum até mesmo por servidores de alta capacidade. O *Mosquitto* implementa o modelo *publilsher/subscriber* e pode ser utilizado em aplicações de *IoT*. Aplicações *IoT* fazem uso de sensores de baixa potência, atuadores, dispositivos móveis, microcontroladores e outros dispositivos programáveis, que podem utilizar mensagens *MQTT* para troca de mensagens (MOSQUITTO, 2022).

O *Mosquitto* oferece comandos de linha, como *"mosquitto_pub"* e *"mosquitto_sub"*, para publicar e subscrever no *broker*, respectivamente, além de bibliotecas em linguagem de programação C para implementação de cliente *MQTT* (MOSQUITTO, 2022).

O *MQTT* é o protocolo de troca de mensagens que atua em conjunto com o *IoT*, criado pela *IBM* no final da década de 90, consequentemente esse protocolo carrega muito do cenário de uso original (MOSQUITTO, 2022). O *MQTT* é voltado e adaptado para sistemas de supervisão e coleta de dados do tipo *SCADA* (*Supervisory Control and Data Acquisition*) (MOSQUITTO, 2022).

Simplificado e eficiente, o *MQTT* é um protocolo com características de segurança, de qualidade de serviço e de facilidade de implementação. Tais características fazem dele um bom candidato para implementações e usos em sistemas embarcados (QUINCOZES; EMILIO; KAZIENKO, 2019).

No padrão *publish/subscriber*, utilizado no *MQTT*, quando um elemento da rede deseja

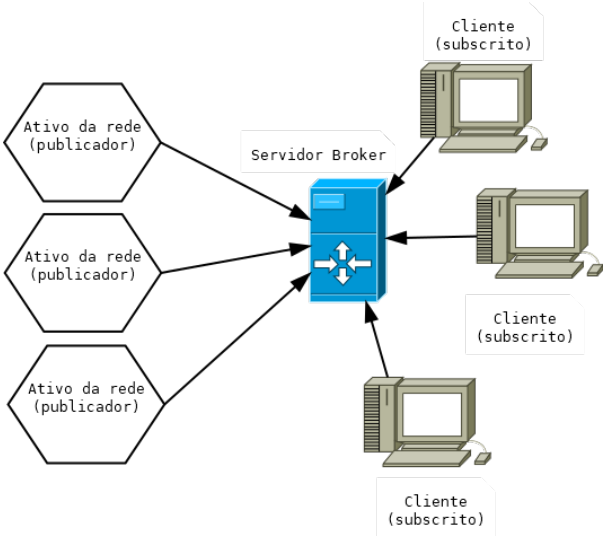

Figura 4 – Funcionamento protocolo *MQTT*. Fonte: Adaptado de (MOSQUITTO, 2022).

receber uma determinada informação, ele a subscreve, fazendo uma requisição para um outro elemento da rede, capaz de gerir as publicações e as subscrições. O elemento da rede *MQTT*, chamado de *Broker*, é o intermediário no processo de comunicação. Outros ativos na rede, que desejam publicar informações, o fazem por meio do *Broker*, enviando as informações que possuem e/ou geraram (MOSQUITTO, 2022).

O *Broker* permite o desacoplamento entre os ativos da rede, não sendo necessária a comunicação direta entre eles. Todavia, isso o torna um elo de fragilidade, pois centraliza as comunicações. Isso não seria possível, por exemplo, em modelos de comunicação do tipo cliente/servidor. Porém ao se implementar um *Broker*, em uma solução de redundância, como em ambiente de computação distribuida, essa fragilidade pode ser mitigada (MOSQUITTO, 2022).

Na Figura 4 é mostrada a visão global do funcionamento do protocolo *MQTT*, onde temos o *Broker* que é o elemento central. De um lado, são mostrados os ativos de rede que desejam publicar no *Broker*, do outro, os clientes, que estão subscritos para receber essas publicações (MOSQUITTO, 2022).

2.4 Aprendizado de Máquina

Segundo Monard e Baranauskas (2003) "Aprendizado de Máquina é uma área de inteligência artificial, cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática". Um programa de computador que toma decisões com base em experiências advindas por meio de soluções bem-sucedidas de problemas anteriores pode ser visto como um sistema de aprendizado. Os sistemas de aprendizado de máquina podem ser classificados quanto à linguagem de descrição, forma de aprendizado utilizado,

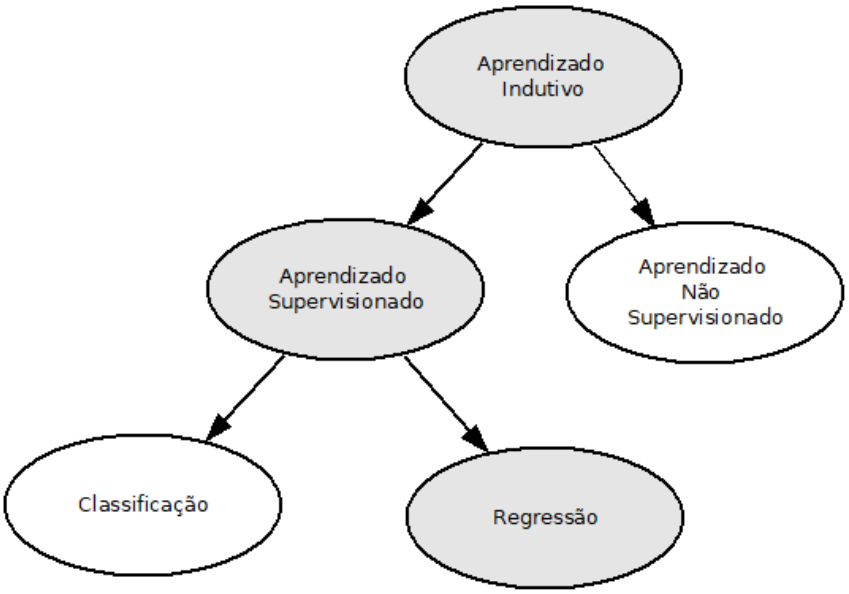

Figura 5 – A hierarquia do aprendizado. Fonte: Adaptado de (MONARD; BARANAUSKAS, 2003).

modo e paradigma, pois possuem características particulares e comuns que os diferenciam (MONARD; BARANAUSKAS, 2003).

De acordo com Monard e Baranauskas (2003) "A indução é a forma de inferência lógica que permite obter conclusões genéricas sobre um conjunto particular de exemplos. Ela é caracterizada como o raciocínio que se origina em um conceito específico e o generaliza, ou seja, da parte para o todo. Na indução, um conceito é aprendido efetuando-se inferência indutiva sobre os exemplos apresentados. Portanto, as hipóteses geradas através da inferência indutiva podem ou não preservar a verdade".

Um dos principais métodos utilizados para adquirir novos conhecimentos e predizer situações futuras é a inferência indutiva. Arquimedes utilizou a indução na descoberta da primeira lei da hidrostática e o princípio da alavanca, Darwin nas leis de seleção de espécies e naturais, Kepler na descoberta das leis de movimento planetário (MONARD; BARANAUSKAS, 2003).

Conforme é visualizado na Figura 5 o aprendizado indutivo pode ser separado em supervisionado e não-supervisionado, os nós sombreados levam ao aprendizado supervisionado utilizando regressão, que foi utilizado nesta dissertação. No aprendizado supervisionado o algoritmo de aprendizado recebe junto com o conjunto de dados de treinamento os seus respectivos rótulos de classe associados. Já no aprendizado não-supervisionado tais rótulos não são disponibilizados (MONARD; BARANAUSKAS, 2003).

O objetivo do algoritmo de indução é construir um classificador que possa de maneira assertiva determinar a classe de novos exemplos dos quais ainda não se conhece os rótulos. Para rótulos de classe discretos, esse problema é conhecido como "classificação"e para valores contínuos como "regressão"(MONARD; BARANAUSKAS, 2003).

A inteligência artificial abrange diversas áreas para simular o pensamento de forma

satisfatória. Segundo Mueller e Massaron (2019) a IA inclui além do aprendizado de máquina:

- **Processamento de linguagem natural:** Ação de possibilitar entrada na linguagem e modificá-la de forma que um computador possa utilizar.
- **Entendimento de linguagem natural:** Ato de interpretar a linguagem para produzir efeito de acordo com o significado a ele atribuído.
- **Representação do conhecimento:** A capacidade de armazenar informações possibilitando acesso rápido a elas.
- **Planejamento (na forma de busca de objetivo):** A capacidade de usar informações armazenadas para tirar conclusões de forma rápida, muito próxima do momento em que acontecem.
- **Robótica:** A capacidade de atuar de acordo com as solicitações de um usuário em alguma forma física.

A base do aprendizado de máquina é a matemática (MUELLER; MASSARON, 2019). Os algoritmos determinam como interpretar grandes quantidade de dados de maneiras específicas. No processamento de dados de entrada os algoritmos criam saídas previsíveis, com base em padrões nos dados fornecidos. A utilização do aprendizado de máquina e da inteligência artificial se deve a necessidade em decifrar os dados de modo a ser possível identificar os padrões que esses contêm e compreendê-los.

Segundo Mueller e Massaron (2019) "tudo no aprendizado de máquina gira em torno de algoritmos. Um algoritmo é um procedimento ou fórmula usada para resolver um problema". Para cada tipo de problema se faz necessário a escolha do tipo de algoritmo, embora a premissa básica seja sempre a mesma: resolver algum tipo de problema. Os problemas podem ser complexos e variados, de toda forma a definição específica do problema levara a solução especifica.

O objetivo de um programa de aprendizado consiste em extrair um bom classificador a partir de um conjunto de dados rotulados. O classificador, pode então ser utilizado para classificar exemplos novos, que ainda não foram rotulados, com a meta de predizer corretamente o rótulo de cada um. Com base nos rótulos obtidos na saída do algoritmo, o classificador pode ser avaliado considerando quanto a sua precisão, compreensibilidade, grau de interesse, velocidade de aprendizado, requisitos de armazenamento, grau de compactação ou qualquer outra propriedade desejável que determine quão bom e apropriado ele é para a tarefa em questão (MONARD; BARANAUSKAS, 2003).

A regressão é uma técnica estatística que tenta definir a relação entre variáveis através de um modelo matemático. A regressão é frequentemente utilizada em problemas onde é necessário definir a causa e efeito entre as variáveis de entrada e os rótulos de saída. A

diferente básica entre problemas de regressão e de classificação está relacionada ao fato de que a primeira o resultado final é um valor numérico, já nos problemas de classificação o resultado é uma classe, uma categoria (JAISWAL; SAMIKANNU, 2017).

2.4.1 Regressão Linear Simples

Segundo Afonso e Nunes (2019) "Existem dois objetivos possíveis e usuais para a utilização da regressão linear. O primeiro é descrever a relação linear entre duas variáveis, onde uma assume o papel de variável independente e outra de dependente; o segundo objetivo é prever valores futuros da variável dependente baseados no valor da variável independente". O modelo de regressão linear simples descreve uma relação linear existente entre uma variável independente, chamada de “x”, e uma variável dependente, chamada de “Y” (AFONSO; NUNES, 2019).

Em um problema estatístico de regressão estuda-se as distribuições de frequências de uma variável para determinados valores fixos de outra variável que está sendo controlada, procurando determinar a maneira de relacionar as variáveis. Em um problema de regressão um objetivo importante é predizer ou estimar o valor mais provável da variável não controlada que corresponde a um valor dado da outra variável (AFONSO; NUNES, 2019).

2.4.2 Aprendizado Supervisionado

Os algoritmos de Aprendizado Supervisionado, quando treinados, recebem exemplos rotulados, logo, cada entrada tem a saída desejada conhecida, o que induz o aprendizado do modelo. Durante o treinamento do modelo, são entregues dois conjuntos de dados, os de entradas, e suas respectivas saídas. Em seguida, o algoritmo realiza a análise da relação existente entre entrada e saída e, como resultado, ele gera um modelo ou uma função, capaz de estimar os rótulos de um novo conjunto de dados de entrada (SÁNCHEZ; COUSO; CASILLAS, 2009) .

Com o algoritmo de AM treinado, são realizados os testes, aqui o modelo gerado é utilizado para estimar valores de saída, a partir de entradas do conjunto de teste. Geralmente esses valores fazem parte do conjunto inicial de dados, e são aleatoriamente separados dos dados que foram utilizados para o treinamento. Por fim, os resultados que foram previstos pelo algoritmo são comparados aos valores reais, possibilitando, assim, uma melhor avaliação do desempenho do algoritmo implementado (FULMARI; CHANDAK, 2014).

No aprendizado supervisionado, a divisão dos dados é realizada em dois conjuntos essenciais, referenciados como conjunto X e conjunto Y. No conjunto X estão a porção dos dados selecionados para análise, já o conjunto Y é formado pelos rótulos relativos aos dados, que são exatamente os “alvos” que se deseja predizer (SÁNCHEZ; COUSO; CASILLAS, 2009). Existe ainda uma divisão no tocante à forma como se manipula os dados: se o rótulo, que é o conjunto Y, é um número real, logo trata-se de uma regressão.

Já se o rótulo pertence a um conjunto finito e não ordenado, trata-se de uma classificação. Esse entendimento é fundamental no momento da escolha do algoritmo a ser implementado (FULMARI; CHANDAK, 2014).

Com um conjunto finito de dados, é escolhido o treinamento por classificação, as entradas, que são o conjunto X, são rotuladas por classificação discretas ou categorias. Assim, o algoritmo busca entender os agrupamentos de dados formados no conjunto de treino, dessa forma constrói-se um modelo baseado no mapeamento do conjunto de treino, que foi recebido, para suas respectivas categorias. Por fim, o algoritmo vincula novas entradas do conjunto de teste a uma ou mais dessas classes que foram geradas, para, assim, realizar o processo de predição (FULMARI; CHANDAK, 2014).

Já no treinamento por regressão, os conjuntos de dados de treino X são rotulados por números reais. Sendo assim, o algoritmo buscará construir um modelo que, dada uma nova entrada com o rótulo ainda não conhecido, seja possível estimar qual será o rótulo dessa entrada, mesmo não o conhecendo previamente (SÁNCHEZ; COUSO; CASILLAS, 2009).

2.4.3 Algoritmo *Random Forest* (RF)

Segundo Breiman (2001), florestas aleatórias (*random forest*) é um algoritmo de classificação composto de diversos classificadores do tipo árvore de decisão, onde cada árvore é composta por um subconjunto de características e de dados de treinamento selecionados aleatoriamente, sendo o resultado da predição é definido por um processo de votação que ocorre entre as árvores do modelo.

Florestas aleatórias podem ser aplicadas a um conjunto de dados com características lineares ou categóricas, nos casos categóricos deve ser transformada em valores entre 0 e $(M - 1)$, onde "M" representa o número de valores que podem ser assumidos pela característica. O uso de florestas aleatórias é bastante efetivo para predição de classes, todavia o desempenho é inferior quando comparado com as regressões. A efetividade das florestas aleatórias está diretamente relacionada a definição das características e instâncias de cada árvore, que ocorrem de maneira aleatória (BREIMAN, 2001).

A floreta aleatória é amplamente utilizada no aprendizado supervisionado tanto para problemas de classificação como de regressão (SHENG et al., 2015) .

O RF é um método multivariado não paramétrico, não é preciso fornecer dados com distribuição normal, além disso é possível trabalhar simultaneamente diversas variáveis, sendo possível um ganho na precisão da classificação e na determinação da importância das variáveis em uma classificação (POLIKAR, 2012).

No algoritmo *Random Forest* são gerados *Decision Trees*, que estabelecem regras para tomada de decisão. Aqui, é criada uma estrutura similar a um fluxograma, com "nós", onde uma condição é verificada e, se atendida, o fluxo segue por um ramo, caso contrário, por outro, sempre levando ao próximo nó até a finalização da árvore. Tendo como base

os dados selecionados para treino, o algoritmo busca as melhores condições e procura a melhor forma de inserir cada uma dentro do fluxo gerado (BREIMAN, 2001).

Floresta aleatória é um algoritmo que se baseia no método estatístico de árvore de decisão, onde a classificação é predita com base em várias variáveis de entrada. Assim como no aprendizado profundo a capacidade de análise é amplificada combinando camadas mais profundas de nós, a floresta aleatória emprega árvores de decisão mais profundas durante o processo de treinamento (POLIKAR, 2012).

De forma divergente ao algoritmo floresta aleatória, o primeiro passo executado nas árvores de decisão é de selecionar aleatoriamente algumas amostras dos dados separados para treino, e não a sua totalidade, como é feito em outros modelos. Quando essa etapa é executada, o *bootstrap*, que é um método de re-amostragem, é utilizado, para selecionar amostras que podem ser repetidas, e é por meio dessa primeira seleção que é construída a primeira árvore de decisão (BREIMAN, 2001).

Na construção de uma árvore de decisão, é preciso definir o primeiro nó da árvore, o nó raiz, que é a primeira condição verificada, dando origem aos dois primeiros ramos da árvore. Por meio da utilização do algoritmo de entropia ou o índice *Gini*, é escolhida a melhor variável para compor o nó raiz, podendo variar de acordo com o método utilizado (TOHRY et al., 2022).

Cada nó na árvore de decisão trabalha em um subconjunto aleatório de informações da base de dados fornecida para o cálculo da saída. Para cada árvore de decisão é gerado uma saída e o resultado é calculado com base na votação majoritária para problemas de classificação ou a média para problemas de regressão (POLIKAR, 2012) ; (SHAHABI et al., 2019).

As floretas aleatórias utilizam a votação para realizar classificação, logo a classe que obtém maior número de votos é selecionada como resultado do modelo. Como a predição é realizada com base e varias árvores de decisão, o erro se torna menos provável de ocorrer quando comparado as árvores de decisão convencionais (MACHADO; MENDOZA; CORBELLINI, 2015).

Nas Figuras 6 e 7 é possível visualizar as fases de treinamento e classificação do classificador de floresta aleatória. Os círculos representam os nós e as linhas representam os galhos em cada árvore de decisão; nós de base correspondem à raiz e são representados por círculos no topo da árvore, enquanto os nós terminais correspondem às folhas e são representados em círculos com contorno amarelo.

Na Figura 6 durante a fase de treinamento cada árvore de decisão no *ensemble* é construída sobre uma amostra aleatória de *bootstrap* dos dados originais, que contém exemplos positivos que estão representados pelos círculos verdes, e exemplos negativos representados pelos círculos vermelhos.

Na Figura 7, durante a classificação, as previsões das classes são baseadas em um procedimento de votação majoritária entre todas as árvores de forma individual. Em cada

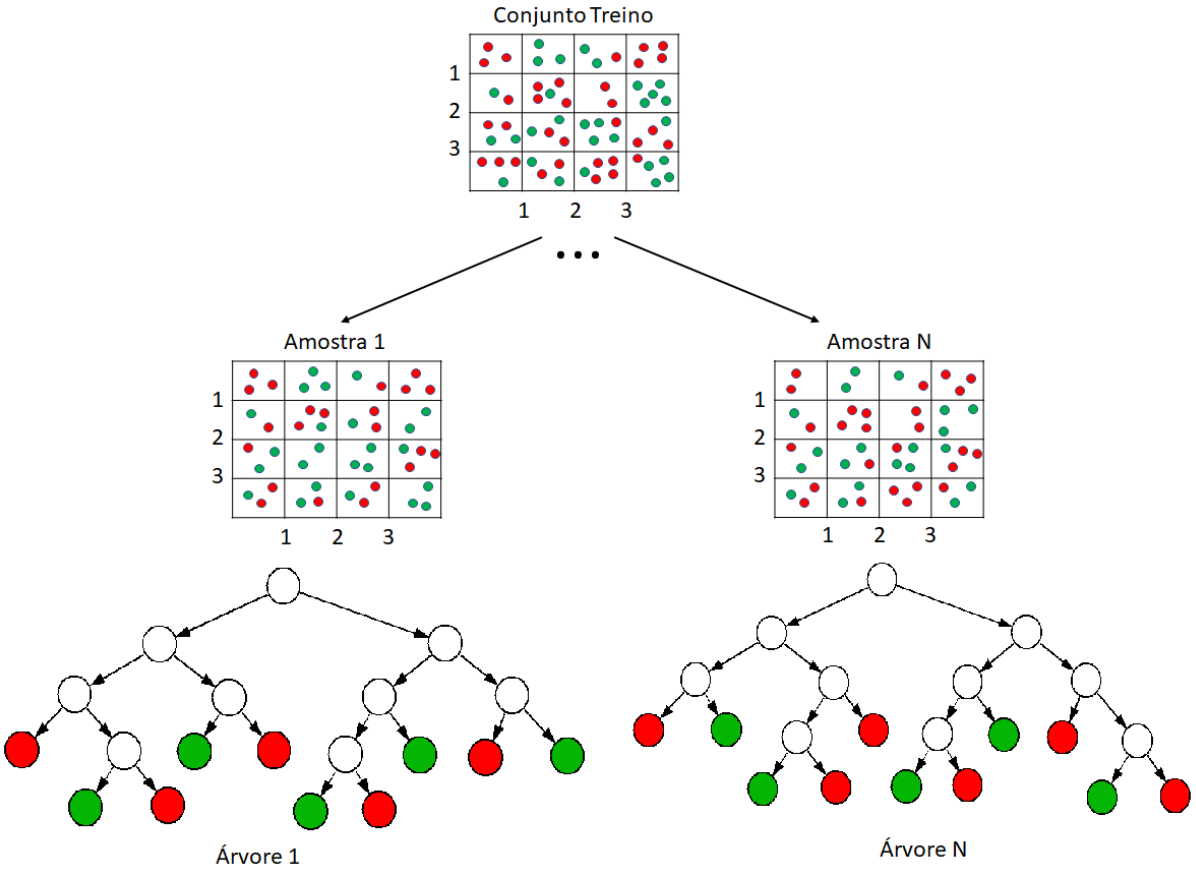

Figura 6 – Fases de treinamento do classificador *RF*. Fonte: Adaptado de (MACHADO; MENDOZA; CORBELLINI, 2015).

uma das árvores, o algoritmo começa no nó raiz na base e percorre a árvore, testando os valores das variáveis em cada um dos nós divididos (azul claro) e, para cada valor, seleciona o próximo ramo a seguir. Até que um nó terminal no topo das árvores seja alcançado este processo é repetido e atribua um rótulo de classe positiva ou negativa à categoria. No final do processo, cada árvore vota no rótulo de classe predito e a classe de saída final é predita (MACHADO; MENDOZA; CORBELLINI, 2015).

Para o algoritmo *Random Forest*, a definição da variável nó raiz não acontece com base em todas as variáveis disponíveis. De forma aleatória, o algoritmo irá escolher duas ou mais variáveis e realizar os cálculos com base nas amostras selecionadas para definir qual dessas variáveis é utilizada no primeiro nó. No processo de escolha da variável do próximo nó, novamente serão elegidas pelo menos duas variáveis, excluindo as já selecionadas anteriormente, e o processo de escolha se repete. Assim, a árvore é construída até o último nó. Durante a criação do modelo, a quantidade de variáveis a serem utilizadas pode ser definida (BREIMAN, 2001).

Fica evidente que esse não é o melhor método para construção de uma árvore de decisão, pois o algoritmo pode, não propositalmente, selecionar as duas piores variáveis na primeira seleção, escolhendo uma variável péssima para o primeiro nó, o que comprometeria toda a árvore. Contudo, como serão construídas muitas árvores, essa estratégia se

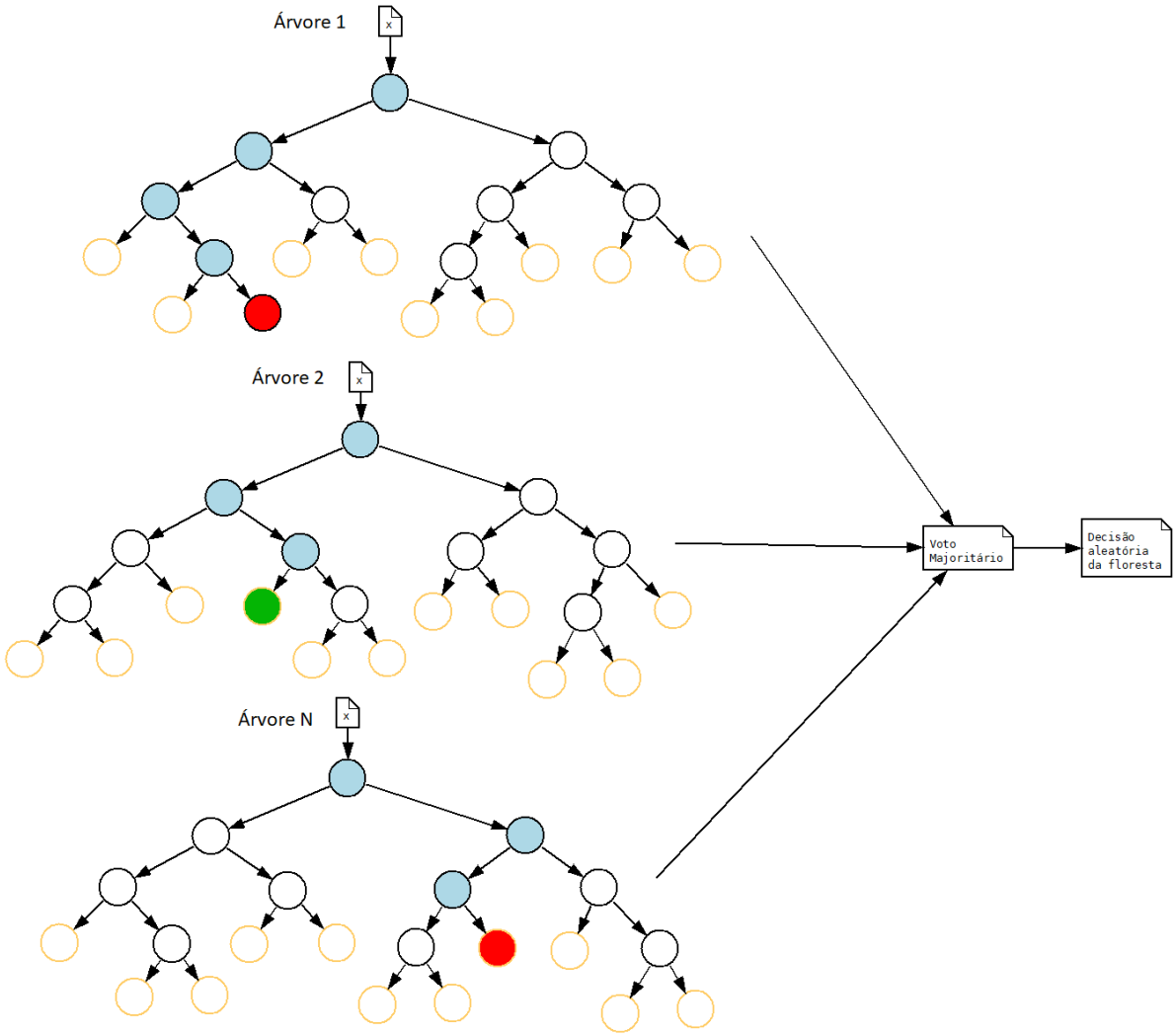

Figura 7 – Fases de treinamento do regressor *RF*. Fonte: Adaptado de (MACHADO; MENDOZA; CORBELLINI, 2015).

torna poderosa, e costuma evitar o sobre ajuste (BREIMAN, 2001).

É possível criar quantas árvores forem desejadas, sendo que quanto mais árvores criadas melhores serão os resultados do modelo, essa definição pode ser aceita, ate determinado ponto, onde uma nova árvore não conseguirá levar a uma melhora significativa no desempenho do modelo.

A criação de novas árvores melhora os resultados do modelo, porem essa definição é valida ate certo ponto, pois existe um momento em que a criação de novas árvores não ira elevar o desempenho do modelo de forma significativa. Vale ressaltar que quanto mais árvores forem criadas maior será o tempo de criação do modelo e consequentemente maior o custo computacional envolvido (BREIMAN, 2001).

Com o algoritmo devidamente desenvolvido e treinado, é possível apresentar novos dados e obter o resultado da previsão, pois cada árvore criada irá apresentar o seu resultado, sendo que em problemas de regressão é realizada a média dos valores previstos, e essa é informada como resultado final. Já em problemas de classificação, o resultado que

mais vezes for apresentado é o escolhido (SHAHABI et al., 2019).

2.5 Transferência de Aprendizado

Existe uma variação da aprendizagem supervisionada, que é o aprendizado por transferência. Enquanto o aprendizado supervisionado é o problema de aprender, uma função ou modelo, que mapeia entradas, e rótulos, com base em pares de exemplos, o aprendizado por transferência é uma variante, que pode ser utilizada quando a tarefa exige um número limitado de exemplos rotulados ou se a escassez de dados não for um problema. Em virtude disso, é possível alavancar o aprendizado por transferência, afim de evitar um grande custo computacional para treinar um modelo que consome grande quantidade de dados (PAN, 2014).

A falta de dados de treinamento pode surgir em casos em que os exemplos rotulados são difíceis ou caros computacionalmente de se coletarem. Em casos como esse, geralmente não há dados suficientes para treinar um modelo em níveis aceitáveis de precisão ou de desempenho desde o início (PAN, 2014).

Um modelo inicial após treinado gera conhecimento que ao ser armazenado e aplicado a um problema diferente, mas que esteja relacionado ao problema original, gera a solução do problema em questão, a um custo computacional mais factível (KUURKOVá et al., 2018).

A fim de contornar a escassez de dados, ou evitar o treinamento de um modelo do zero, o aproveitamento do conhecimento adquirido em uma tarefa relacionada à tarefa original, para a qual existem muitos exemplos rotulados, pode resolver a tarefa em questão. Dessa forma, é possível dizer que esse é o conceito principal do aprendizado por transferência, e geralmente é bem-sucedido quando as tarefas de origem e de destino exigem informações semelhantes para serem resolvidas (KUURKOVá et al., 2018).

Nessa dissertação as amostras de dados utilizada para o treinamento do algoritmo de inteligência artificial estão disponíveis nos locais de monitoramento, por exemplo, barragens em locais afastados, contudo, temos limitação em movimentar todo este volume de dados a uma central de monitoramento. Por esta razão, a dissertação aborda o treinamento do modelo nos locais de monitoramento e movimenta o modelo à central de monitoramento, evitando a movimentação de grandes volumes de dados.

Capítulo 3

Trabalhos Relacionados

Nesta seção serão discutidos alguns trabalhos que são correlatos ao trabalho proposto nesta dissertação, que envolvem um ou mais tópicos que estão relacionados ao tema central, ou que se assemelham às tecnologias empregadas na solução do problema.

Na Tabela 2, são apresentados os comparativos dos trabalhos correlatos com a presente dissertação, destacando os pontos de maior semelhança entre eles. As características destacadas rementem aos pontos de destaque dessa dissertação conforme abaixo:

- **Contribuições ao projeto *ADMITS*:** Relaciona se o trabalho analisado tem alguma ligação com o projeto *ADMITS*, e se contribui de alguma maneira para o avanço e divulgação do trabalho.

- **Internet das coisas (*IoT*):** Analisa se o trabalho utiliza ou aborda a utilização da tecnologia *IoT* na construção ou elaboração da solução.

- **Simulação a cenários de desastres:** Verifica se o trabalho analisado realiza simulações referentes a cenários de desastres, ou relacionados a este escopo.

- **Viabilidade em locais remotos:** Analisa se o trabalho propõe ou investiga a viabilidade da solução em locais afastados, zonas remotas, ruais, ou com carência em recursos de infraestrutura de TIC.

- **Uso de aprendizado de máquina:** Verifica se o trabalho analisado faz uso de inteligencia artificial no tocante a aprendizado de máquina na solução apresentada.

- **Transferência de aprendizado:** Analisa se o trabalho faz uso de transferência de aprendizado na solução proposta pelo autor.

- **Construção da arquitetura:** Verifica se o trabalho analisado construiu a arquitetura para realização de experimentos, ou se utilizou de alguma ferramenta já existe para tal.

Tabela 2 – Comparativo Trabalhos Correlatos

Características dos trabalhos	1	2	3	4	5	6	7
Contribuição ao projeto *ADMITS*						x	
Internet das Coisas *IoT*	x	x		x	x	x	
Simulação a cénários de desastre	x	x	x	x		x	
Viabilidade em locais remotos		x	x			x	x
Uso de Aprendizado de Máquina	x	x	x	x	x	x	x
Transferência de aprendizado		x	x				
Construção da arquitetura				x	x		x

Tabela 3 – Relação do Número do trabalho correlato à referência bibliográfica

N° Trabalho Correlato	Referência bibliográfica
1	(FURQUIM et al., 2017)
2	(SACCO et al., 2020)
3	(BRUNEAU; TAMISIER, 2019)
4	(SHEIBANIFARD; SHAHROOD; POUYAN, 2021)
5	(SILVA, 2019)
6	(PASQUINI et al., 2020a)
7	(CUNHA et al., 2019)

A Tabela 3 relaciona o número do trabalho correlato que consta na Tabela 2 com as respectivas referências bibliográficas. É notável que o uso de aprendizado de máquina está presente em todos os trabalhos analisados, e o uso de *IoT* tem sido bastante abordado, já a transferência de aprendizado não é tão abordada, bem como a construção da arquitetura. Por fim, as contribuições ao projeto *ADMITS* são escassas devido ao projeto estar em uma fase ainda inicial.

A seguir cada um dos trabalhos presentes na Tabela 2 será apresentado.

3.1 Identificação de Desastres Usando IoT

A identificação de cenários de desastres, é uma das principais abordagens do projeto ADMITS. No trabalho referenciando em (PASQUINI et al., 2020b), são listados os propósitos que compõem o *ADMITS* as tecnologias envolvidas, e todas as colaborações que a integração da fase atual de *TIC* trazem. O trabalho é uma apresentação completa do projeto ADMITS, e de como os recursos oferecidos por *IoT* compõem soluções para monitoramento. O *IoT* trabalha com grande número de ativos de rede, o que é fundamental para identificação de cenários de desastre. Esses conceitos estão sendo aplicados neste trabalho.

Na Figura 8, é apresentado o ecossistema do projeto ADMITS. Na base, estão os desastres, como queimadas, enchentes, chuvas fortes, descargas elétricas e eletromagnéticas, desabamentos, terremotos, dentre outros.

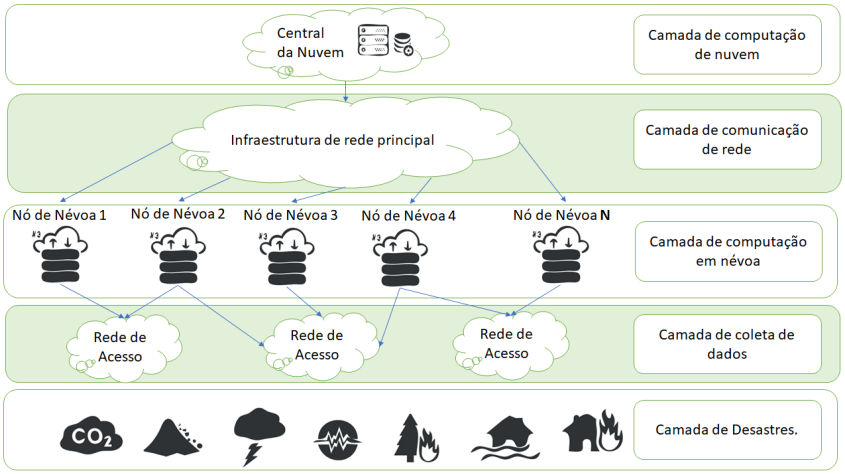

Figura 8 – Ecossistema projeto ADMITS, Fonte: Adaptado de (PASQUINI et al., 2020b).

Na camada superior, estão os dispositivos de *IoT* necessários para o monitoramento do ambiente, como sensores de temperatura, umidade e, de forma geral ativos de sensoriamento da rede.

A próxima camada é composta pela computação em névoa, que são os "nós" da rede, que realizam a computação necessária, no processamento das informações coletadas na camada inferior. Subindo para a próxima camada, está a comunicação de rede, que é a interligação lógica e física entre as névoas de computação da camada anterior.

Em Sheibanifard, Shahrood e Pouyan (2021), é realizado um trabalho em que se utiliza a integração de computação em nuvem, de computação de borda e de névoa em uma mesma arquitetura com dispositivos IoT. Os autores propuseram elevar a robustez da infraestrutura para gerenciamento das situações de desastres.A arquitetura proposta pelos autores apresenta uma redução no consumo de energia e na latência de comunicação de forma significativa. Ao final do trabalho foi proposto como projeto futuro implementar a tolerância a falhas na infraestrutura proposta, e a presente dissertação apresenta uma solução com tolerância a falhas em cenários de desastre.

3.1.1 Aplicação de Aprendizado de Máquina na previsão de desastres

O trabalho realizado em FURQUIM et al. (2017) aborda o uso de redes sem fio para realização de monitoramento em centros urbanos. As simulações são abordadas com o SENDI, que é um sistema tolerante a falhas baseado em *IoT* e em aprendizado de máquina para detecção e previsão de desastres.

Para viabilizar esse trabalho, foi feito um estudo de caso sobre previsões de enchentes. Com os resultados obtidos nos experimentos realizados, os autores concluíram que o SENDI permite a geração de alertas para tomada de decisão em tempo hábil. Todavia, foi demonstrado também que a acurácia sofre variação, dependendo do nível de degradação

da infraestrutura. A presente dissertação demonstra que ao aplicarmos transferência de aprendizado na realização de previsões de desastres, é possível obter acurácia equivalente ao modelo padrão, mesmo em cenários de desastre.

Em Sacco et al. (2020), é apresentado um trabalho em que é proposta uma solução para gerenciamento de tarefas em sistemas de monitoramento de desastres com uso de uma rede de dispositivos *IoT*. Os autores exploram o uso de aprendizado profundo para detecção de vozes humanas em cenários de desastres. Nesse trabalho, foi desenvolvido um simulador em linguagem *C++* orientado a eventos. De forma similar, a presente dissertação também realizou a construção da infraestrutura de experimentos em uma rede com dispositivos *IoT*. Todavia, está dissertação apresenta a viabilidade de manter o monitoramento diante das falhas na infraestrutura, sofridas nos cenários de desastre.

3.1.2 Aplicação de Aprendizado de Máquina para Estimativas de Métricas de rede

No trabalho referenciado em Cunha et al. (2019), é realizado um estudo de um serviço de armazenamento de dados não relacional, buscando a construção de um mecanismo que consiga estimar o valor de métricas de qualidade de serviço. O trabalho realizou experimentos como a introdução de padrões de carga no serviço e a observação da oscilação causada na qualidade de serviço recebida por um cliente, que procurou predizer uma métrica de *QoS* que um cliente receberia. Por fim, esse trabalho demonstrou que o efeito causado pelas variações de carga é perceptível a partir da observação de um cliente e que é possível criar um modelo que estime valores de métricas de *QoS* a partir de estatísticas de infraestrutura. A presente dissertação amplia as investigações ao adotar a transferência de aprendizado no contexto de cenários de desastres que apresentam, potencialmente, severas restrições de comunicação.

O aprendizado de máquina pode ser aplicado de diferentes formas, dentre elas está a geração de estimativas para métricas de rede. No trabalho referenciado em(CUNHA et al., 2019), é realizado um estudo em torno de um serviço de armazenamento de dados não relacional, buscando a construção de um mecanismo que consiga estimar o valor de métricas de qualidade de serviço.

Já o trabalho em questão utilizou estatísticas da infraestrutura do serviço como base e realizou a implementação de algoritmos de inteligência artificial para realizar predições. A abordagem seguiu a linha de identificação em tempo hábil e contramedidas preventivas. Também houve a implantação de um ambiente de testes para simular interações entre o serviço e os clientes.

O trabalho realizou experimentos como a introdução de padrões de carga no serviço e a observação da oscilação causada na qualidade de serviço recebida por um cliente. Com os conjuntos de dados necessários extraídos, um método de aprendizado de máquina

foi utilizado para gerar um modelo que conseguisse predizer os valores de uma métrica de qualidade de serviço que um cliente receberia. Os autores concluíram que é possível reduzir o conjunto de estatísticas utilizadas no treinamento do modelo de aprendizado de máquina, sem que haja degradação na qualidade da estimativa.

A presente dissertação aplicará conceitos semelhantes, construindo uma infraestrutura para experimentos, e desenvolvendo os algoritmos de aprendizado de máquina e o uso da transferência de aprendizado. Todavia, a presente dissertação abordará um cenário de desastres, trazendo maior relevância para solução, contribuindo, assim, com o projeto *ADMITS*.

3.1.3 Plataformas Computacionais de Simulação para Cenários de Desastres

O trabalho desenvolvido em REIS (2015) trata sobre enchentes e inundações urbanas e as razões para tais eventos. Esse trabalho utiliza-se de ferramentas de análise e de simulação para modelagem dos cenários de desastre. São realizadas simulações de manchas de risco de inundação em microrregião, e é proposta a avaliação de diferentes modelos desse tipo de modelagem. Os autores concluíram que é possível a criação de mapas de riscos a enchentes e inundações e, também, que a tomada de decisões deveria ser realizada de forma unificada, utilizando dois modelos de simulação. Em comparação com a presente dissertação, existe a semelhança quanto à realização de experimentos relativos a cenários de desastres, no entanto, nesta dissertação, construímos nossa própria infraestrutura para os experimentos (emulador) em vez de utilizarmos simuladores.

Dentre os desatares naturais que afetam países de todo o mundo, o impacto da inundações foi tema abordado no trabalho descrito em (BRUNEAU; TAMISIER, 2019). Os autores estudam o uso de redes neurais profundas como mecanismo de identificação em imagens multimídia, para identificar o impacto das inundações em desastres.

No trabalho Bruneau e Tamisier (2019), é proposto um classificador de imagens que tem como base o nível de impacto gerado na inundação. Também foi realizado pelos autores uma comparação entre modelos de aprendizado profundo. Com a realização desses experimentos, foi concluído que o modelo *MobileNet* resultou em maior precisão que os demais testados.

Os autores identificaram dificuldade em definir a gravidade da inundação sem conhecimento prévio, e que esse processo é demorado. A presente dissertação também realiza experimentos quanto a cenários de desastre, aplicando o aprendizado de máquina na realização de previsões. No entanto, nesta dissertação, o cenário se apresenta mais amplo, com uma rede de dispositivos *Iot*, com construção da arquitetura de experimentos e com contribuição para o projeto *ADMITS*.

Capítulo 4

Proposta da Dissertação

O projeto *ADMITS* busca, por meio do uso de tecnologias e da rede mundial de computadores, usufruir dos benefícios de *IoT* para monitorar e manter sistemas de prevenção a desastres, de modo a prever esses incidentes, com certa antecedência, o que permitiria salvar muitas vidas e, em alguns casos, até mitigar o desastre. Esse capítulo apresenta a proposta de uma arquitetura para viabilização do monitoramento a cenários de desastre. A arquitetura apresentada utiliza frações dos dados coletados pelos dispositivos de sensoriamento de modo a evitar a geração de fluxos massivos de dados na infraestrutura. O uso da transferência de aprendizado dos modelos de AM gerados nos locais de monitoramento e enviados a central terá papel habilitador na solução proposta.

4.1 Visão Geral

O cenário exibido na Figura 9 apresenta a interação entre os elementos da arquitetura em questão, conforme descritos abaixo.

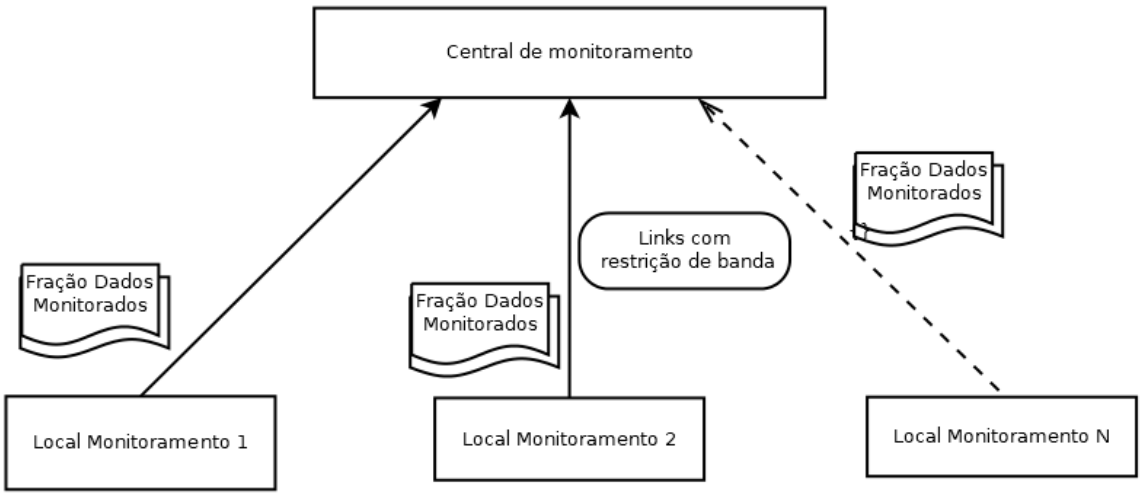

Figura 9 – Visão geral da arquitetura. (Fonte: Do autor (2022)).

- **Central de Monitoramento:** Ponto central de monitoramento de desastres para um conjunto N de locais monitorados. Na central de monitoramento são recebidos os modelos de AM treinados nas localidades monitoradas, bem como uma fração das amostras coletadas nestas localidades, visando acompanhar a situação das localidades remotas. Opcionalmente, os modelos recepcionados e armazenados na central de monitoramento podem ser atualizados, utilizando-se para tanto, as frações de amostras recebidas.

- **Locais de Monitoramento:** Locais onde de fato o monitoramento é realizado, a partir de inúmeros sensores IoT. Pode ser, por exemplo, uma barragem. No local de monitoramento utiliza-se uma rede local para monitoramento dos dados IoT e, a partir de um ambiente local de processamento, modelos de AM são treinados e atualizados. Estes modelos treinados localmente são serializados em direção a central de monitoramento, bem como um pequeno conjunto de amostras.

- **Comunicação entre a central e locais de monitoramento:** A comunicação entre a central e os locais monitorados se da através da Internet. Os locais de monitoramento são zonas afastadas com limitações nos enlaces de dados que as interligam. Serão abordadas diferentes formas de realização desta comunicação.

A central de monitoramento, onde são concentradas as informações coletadas, que são divididas em dados de sensoriamento e de modelos treinados de aprendizado de máquina, em cada local de monitoramento. Esse trabalho pode ser escalado para vários pontos de monitoramento. Esses dados obtidos pelos dispositivos integrantes da rede IoT são fornecidos para realizar o treinamento do algoritmo, sendo parte deles utilizados para treino, e outra parte para testes, conforme descrito na Seção 5.1. Uma vez que o treinamento está sendo feito localmente nas respectivas cidades de monitoramento, esse pode ser aproveitado e enviado à central de monitoramento, que aproveita o modelo treinado e utiliza conjuntos de dados fracionados e não com a sua totalidade.

Como mostrado na Figura 12, os enlaces de dados, que interligam as cidades de monitoramento a central, têm restrição de largura de banda, além de baixa qualidade de *QoS (Quality of service*), de ponta a ponta, dificultando ainda mais a solução do problema.

São investigados dois cenários para implementação da transferência de aprendizado, no primeiro os modelos de AM treinados localmente são transferidos a central juntamente com o menor conjuntos de dados proposto, que por sua vez consome o mínimo os recursos de infraestrutura. Em seguida são calculados os *NMAEs*, para todos os modelos enviados, e comparados com o modelo padrão.

No segundo cenários são enviados os modelos de AM treinados, a central, juntamente com todos os conjuntos de dados propostos. Também são realizados os cálculos dos *NMAEs* afim de demonstrar a efetividade da transferência de aprendizado no monitoramento a cenários de desastre.

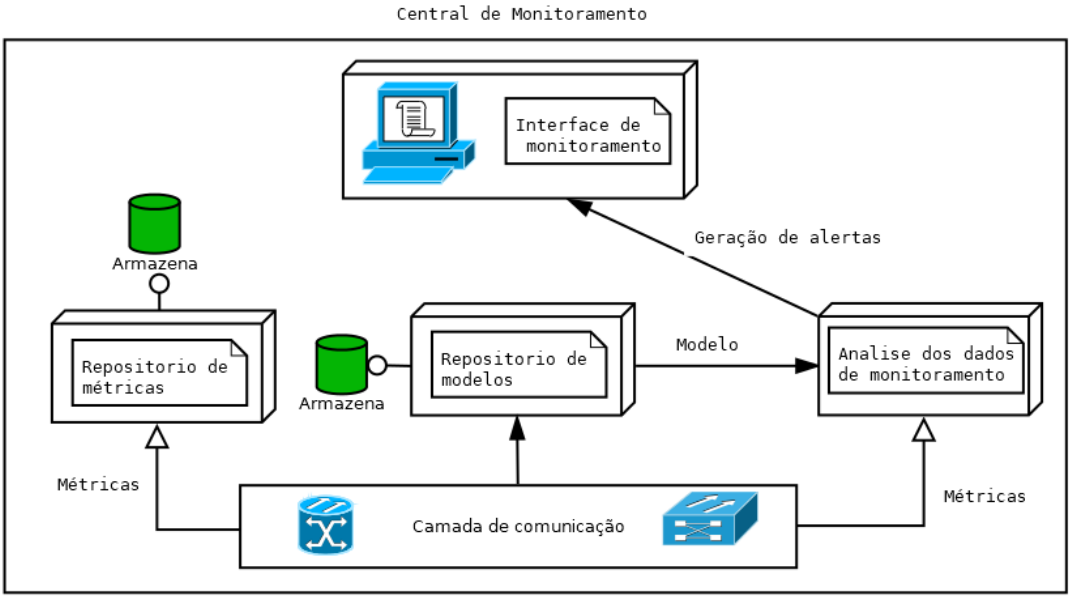

Figura 10 – Arquitetura em camadas da central de monitoramento. (Fonte: Do autor (2022)).

Para verificar os modelos de aprendizado de máquina criados, é utilizado o erro médio absoluto normalizado, chamado de *NMAE*, utilizado em modelos de regressão. Uma vez que o volume de dados foi drasticamente reduzido, quando comparados a uma leitura que é feita a cada um segundo, chamada aqui de modelo padrão, é esperado que exista uma discrepância entre o *NMAE* da proposta da dissertação, conforme está descrito na Seção 5.3. Encontrar pouca discrepância na comparação do *NMAE* do modelo padrão, frente à proposta da dissertação, indica que os objetivos descrito na Seção 1.2 foram alcançados.

4.1.1 Arquitetura da central de monitoramento

Na central de monitoramento são recebidas frações dos dados e os modelos que foram treinados nos locais monitorados, pela camada de comunicação. Os modelos são armazenados na central e utilizados para realizações das predições de desastre. Na camada de análises de dados de monitoramento, os conjuntos de dados recebidos são fornecidos aos modelos armazenados para realização das previsões de desastre. De acordo com a Figura 10 a camada de interface de monitoramento recebe os alertas gerados pela camada de análise de dados de monitoramento.

4.1.2 Arquitetura dos locais de monitoramento

Nos locais de monitoramento, na camada mais baixa, estão os dispositivos de sensoriamento que realizam as leituras de diversas métricas quanto a cenários de desastre. Na camada de interconexão de rede acima estão os ativos de rede que realizam as interconexões entre esses dispositivos e o servidor local de monitoramento. Na camada do servidor

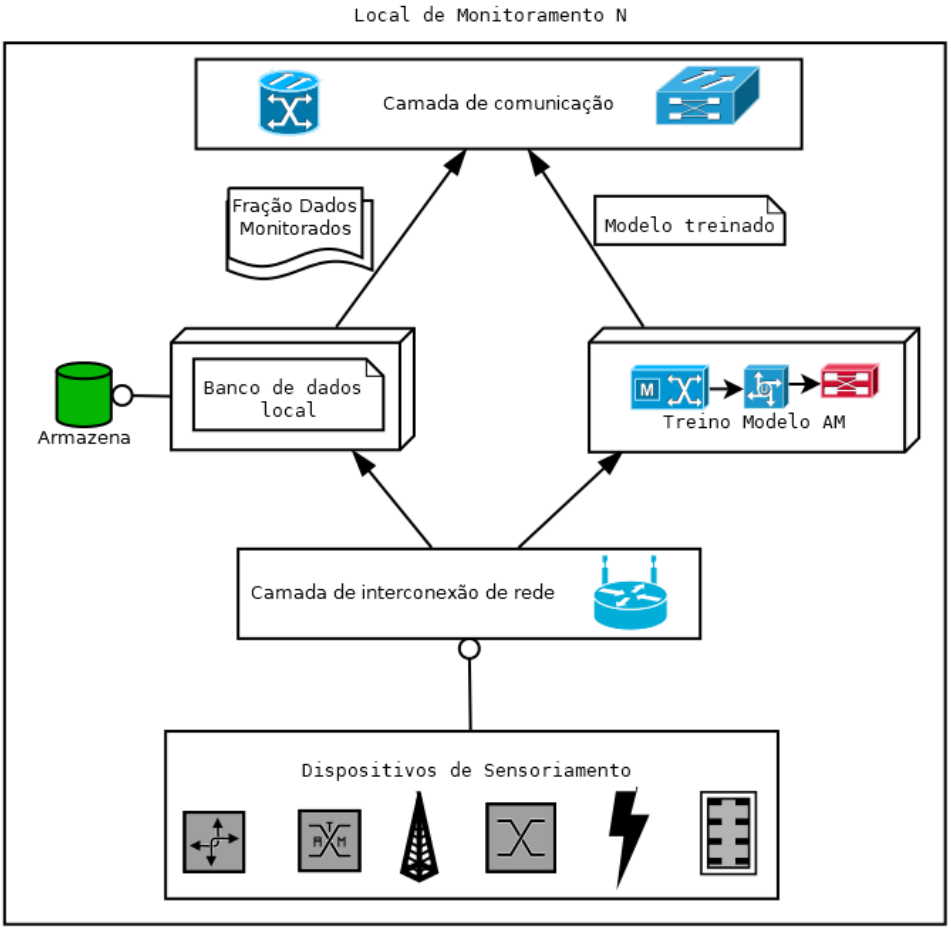

Figura 11 – Arquitetura em camadas dos locais de monitoramento. (Fonte: Do autor (2022)).

local onde os dados são armazenados em sua totalidade e utilizados para o treinamento do modelo de aprendizagem de máquina. Conforme é possível visualizar na Figura 11, o modelo treinado, bem como os conjuntos de dados coletados, são enviados para a central usando a camada de comunicação.

4.2 Implementação da arquitetura

Toda a infraestrutura construída foi modelada virtualmente com uso de máquinas virtuais, enlaces de *hypervisor* e contêineres *Docker*. Nesse ponto, foi preciso realizar um estudo sobre a latência real existente entre os enlaces de dados das cidades de monitoramento para a central. Para tal, foram realizados experimentos de disparos de pacotes entre as respectivas cidades, conforme descrito na Seção 5.1, em que foi possível desenhar um padrão de comportamento na latência de comunicação existente entre esses enlaces de dados. Com o processamento desses dados, que consiste na realização de alguns cálculos matemáticos descritos na Seção 5.1, foi possível mensurar o real atraso de comunicação entre as cidades de monitoramento.

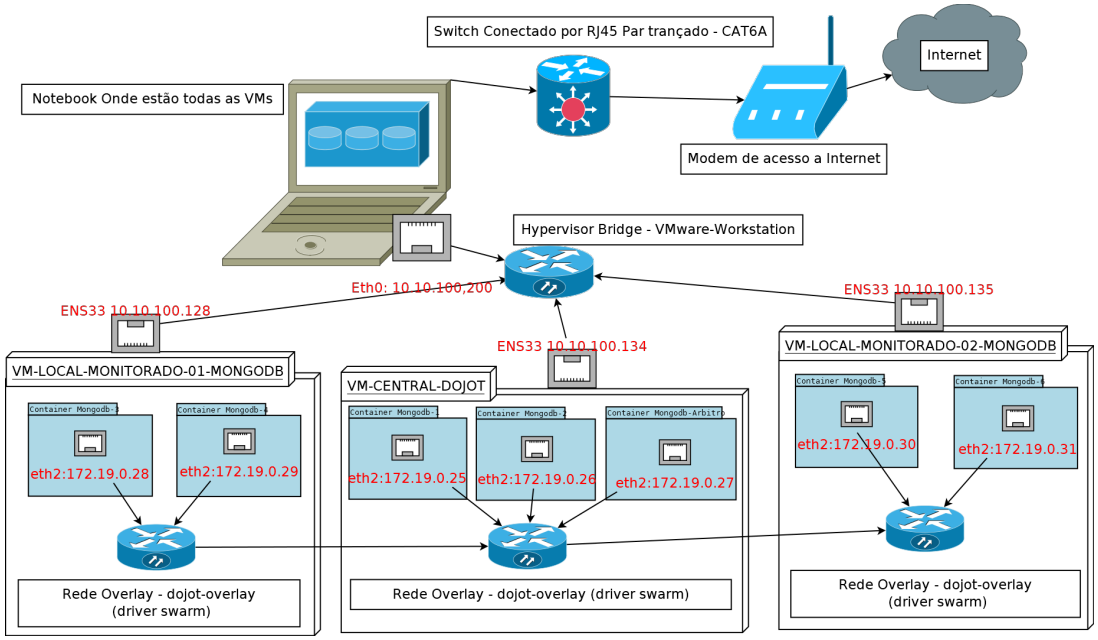

Figura 12 – Infraestrutura Virtualizada. (Fonte: Do autor (2022)).

Com os valores já conhecidos de atrasos entre os enlaces de dados existente entre as cidades de monitoramento e a central e suas respectivas variações, um escopo de dados da variação na latência de comunicação foi criada. Esse escopo foi utilizado pelo software *Traffic Control TC*, um programa de código aberto do Linux, que permite a configuração de latência entre interfaces de rede. Assim, foi configurado um atraso entre os enlaces de dados das máquinas virtuais, de modo a reproduzir o real cenário de comunicação existente entre os locais monitorados e a central.

Na Figura 12, é apresentada a infraestrutura virtualizada que foi construída. Aqui existem três máquinas virtuais para representação de cada uma das cidades de monitoramento e suas respectivas centrais, como também a concentração dos dados locais advindos da rede de dispositivos de monitoramento *IoT*, além de hospedar os contêineres *docker*. E, nesses contêineres, estão sendo executadas as instâncias que compõem o *dojot*, e todas as soluções que o compõem descritos na Seção 2.3.1. O destaque aqui vai para o banco de dados *MongoDB*, que foi implementado de uma forma diferente ao convencionalmente utilizado em instâncias *dojot*. Ele foi colocado em modo distribuído, de forma que os contêineres utilizados para prover o serviço do *MongoDB* estão sendo executados em diversos locais, a fim de trazer maior redundância das informações armazenadas e uma maior disponibilidade. Dessa forma, caso ocorra falha ou interrupção em alguns dos contêineres que compõem o *cluster*, os demais são capazes de persistir os dados sem interrupção da instância *dojot*, ficando totalmente transparente aos usuários, que não perceberiam se algum contêiner sofresse interrupção.

Para a clusterização do banco de dados, foi utilizado o driver *Swarm* do docker, que é a tecnologia do orquestrador de contêiner, responsável por manter e disponibilizar uma rede *overlay*, que é uma rede sobreposta sobre outra rede, de forma a prover comunicação

entre os contêineres que compõem o *cluster*, utilizando outra faixa de endereços *IP*, com maior controle e segurança na troca de mensagens entre os contêineres.

A rede que interliga as maquinas recebeu o prefixo 10.10.100.x. Como cada máquina virtual possui uma interface de rede para comunicação externa e acesso à Internet, todas podem se comunicar por esses enlaces com o nome padrão de ENS33. Para as máquinas virtuais, a interface de nome ETH0 é o *driver* físico de rede do computador que as hospeda, conforme mostrado na Figura 12. Já os contêineres, que estavam utilizando o *driver swarm* do docker, estão em uma rede de prefixo 172.19.0.x, e o nome das interfaces de rede é ETH2 em todos os contêineres.

4.3 Comunicação entre os elementos da arquitetura

Como as máquinas virtuais estão hospedadas em um mesmo computador físico, e cada uma possui uma interface de rede ENS33 que opera com faixa de *IP* externa ao *hypervisor*, a latência entre elas era de menos de 1ms (milissegundo), o que, apesar de parecer muito bom, não representa a realidade dos enlaces de dados entre as cidades de monitoramentos. A fim de reproduzir o cenário de forma realista, o software *TC* foi configurado nas interfaces de rede das máquinas virtuais, de modo a reproduzir os atrasos de latência, que realmente existem entre essas cidades.

O *TC* está implementado no *kernel* do Linux, e possui um conjunto de algoritmos para controle de tráfego. Esses algoritmos permitem modificar a forma como os pacotes recebidos pelo sistema operacional são encaminhados pela rede. O *TC* realiza o controle de tráfego por meio de dois mecanismos:

- Os pacotes são policiados na entrada, e, por meio desse policiamento, os pacotes indesejáveis são descartados;
- Os pacotes são enfileirados na respectiva interface da rede de saída, e esses pacotes armazenados nas filas podem ser atrasados, marcados, descartados ou priorizados.

Em sistemas operacionais com *Kernel* Linux, as políticas de *QoS* são criadas de forma independente para cada interface do equipamento. O policiamento é feito para pacotes encaminhados a uma determinada interface, e o controle de tráfego, propriamente dito, para os pacotes que são enviados pela interface. Por exemplo, em um roteador, cada interface controla o *QoS* do tráfego em um único sentido. Dessa forma, se um roteador possui uma interface para a *LAN* interna e outra *WAN* para Internet, então as políticas de *QoS* na interface *LAN* controlam o que é transmitido para Internet, e. na interface WAN, o que é recebido.

O *TC* possui funcionalidades que podem ser acessadas através de linha de comando, o que permite criar políticas de *QoS*, que definem a sequência de algoritmos que são aplicados para tratamento do tráfego. A política de *QoS* é definida cascateando-se elementos

lógicos, que representam por quais etapas os pacotes serão encaminhados. Na tratativa realizada pelo TC, antes que os pacotes sejam entregues à interface de saída, eles são analisados pelos algoritmos do TC, que realizam as tratativas que podem variar de acordo com as configurações inseridas.

Por fazer parte do *Kernel* Linux, o *TC* pode utilizar as ferramentas para simular atraso e perda de pacotes para aplicativos *UDP* ou *TCP* ou limitar o uso de largura de banda de um serviço específico, a fim de simular conexões de Internet, como *DSL*, Cabo, T1, dentre outros (TC, 2022).

Uma vez que as comunicações entre as máquinas virtuais ocorre através de um *switch*, que entrega um atraso abaixo de um milissegundo, foi preciso inserir um atraso na comunicação. O *TC* é quem possibilita essa configuração, porém, como na prática os atrasos não são fixos, e sim variantes com o tempo, o *TC* também realiza uma distribuição na inserção desses atrasos ao longo do tempo. Para isso, ele possui algumas listas de distribuição como; distribuição normal, distribuição pareto e distribuição pareto normal.

Entretanto, nenhuma das distribuições disponíveis nativamente no *TC* conseguiu representar o real cenário de atraso entre os enlaces de dados entre Uberlândia e Mariana, e Uberlândia e Brumadinho respectivamente. Assim sendo, para aferir a real latência entre os enlaces de comunicação entre as cidades propostas, foram realizados experimentos que estão descritos no Capitulo 5. Com base nesses experimentos foram geradas amostras que compõem um arquivo de distribuição para utilização do *TC* na inserção das variações de atraso na comunicação. Com esses arquivos, o *TC* realiza a inserção dos atrasos de comunicação entre as interfaces de rede, o que possibilita a reprodução do cenário real no protótipo.

4.4 Modelos de aprendizado de máquina e transferência de conhecimento

Foi implementado um modelo de aprendizado de máquina para detectar desastres usando dados que são obtidos dos locais de monitoramento a cada segundo. Esse modelo será chamado de “modelo padrão” ao longo desta dissertação. A presente proposta envolve a realização da leitura de monitoramento em diferentes janelas de tempo: 1, 30, 60, 120 e 300 segundos os intervalos de envio dos dados. Cada um dos modelos criados à partir de tais janelas de tempo será chamado de “proposta da dissertação”.

O foco é investigar o que acontece quando somente uma parcela dos dados coletados pelos locais de monitoramento são enviados para a central, conforme é visualizado na Figura 13. A proposta busca uma solução que seja habilitadora a cenários de desastres em regiões remotas, logo o consumo de fluxos massivos de dados pela infraestrutura é um fator observado. São abordados diferentes cenários para o envio das mensagens entre local monitorado e central de monitoramento, com diferentes intervalos de tempo.

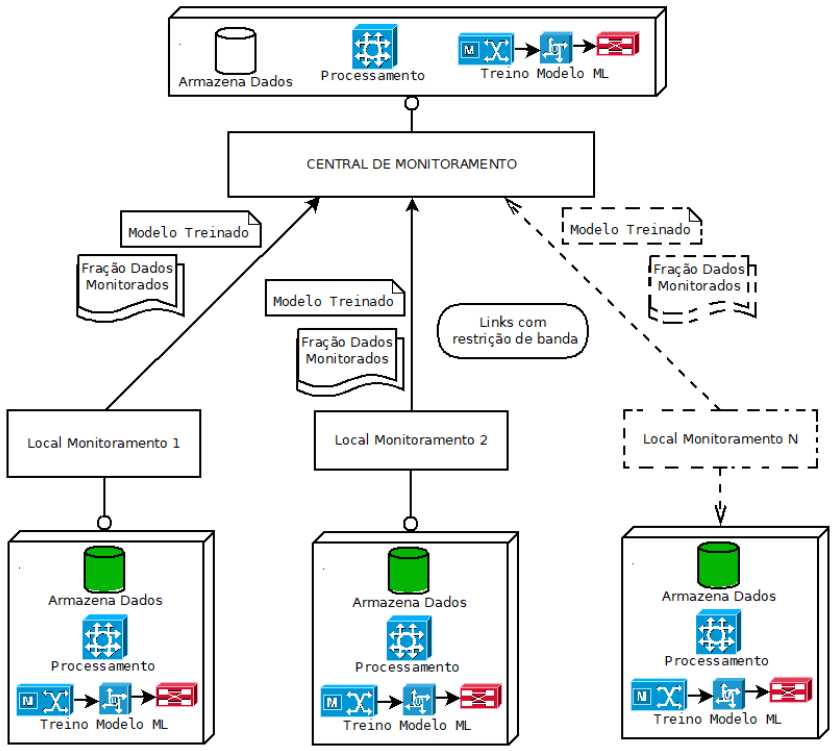

Figura 13 – Visão global da arquitetura com transferência de aprendizado. (Fonte: Do autor (2022)).

No Capítulo 5, que se segue, são detalhados os experimentos que foram realizados para validar as hipóteses levantadas na Seção 1.3, e assim demonstrar a contribuição da proposta para o avanço no estado da arte.

Capítulo 5

Experimentos e Análise dos Resultados

Conforme descrito no Capitulo 4, foi construída uma infraestrutura virtualizada para hospedar uma instância do *dojot*, com banco de dados *MongoDB* em modo distribuído. Uma máquina virtual foi criada para representar cada cidade proposta no monitoramento. Este Capítulo é dedicado à descrição dos experimentos realizados, aos experimentos utilizados na configuração da infraestrutura, ao desenvolvimento dos algoritmos de aprendizado de máquina e à transferência de aprendizado, de modo a comprovar as hipóteses levantadas na Seção 1.3. Também serão apresentados todos os gráficos e os resultados obtidos nos experimentos, bem como a análise empírica das informações geradas nos experimentos. A Tabela 4 lista os experimentos que foram realizados.

5.1 Experimentos

5.1.1 Aferindo latência entre Uberlândia e Brumadinho

Para realização dos experimentos foi construída uma infraestrutura virtualizada que está descrita na Seção 4.2, contudo este primeiro experimento se fez necessário para a configuração dos atrasos de comunicação nos enlaces de dados da infraestrutura. O enlaces devem reproduzir o real atraso na comunicação existente. Para tal, foi feito um experimento afim de verificar este atraso. Primeiramente, foi realizada o disparo de pacotes utilizando o comando *PING* da máquina virtal chamada de "VM-UDIA-DOJOT"que é a central em Uberlândia, e foram disparados 95000 pacotes *ICMP (Internet Control Mes-*

Tabela 4 – Experimentos Realizados

Experimento	Descrição
1	Aferindo latência entre Uberlândia e Brumadinho
2	Aferindo latência entre Uberlândia e Mariana
3	Realizando predições com algoritmo de AM
4	Mensurando consumo de tráfego na infraestrutura

sage Protocol). O disparo gastou um tempo próximo de 24 horas para ser executado, com destino à prefeitura de Brumadinho, e os resultados foram armazenados em um arquivo de *log*.

Para o disparo dos pacotes *ICMP* da máquina virtual na central em Uberlândia, para a prefeitura de Brumadinho, foi utilizado o comando *"PING"*, juntamente com o parâmetro "-c", que define o número de disparos serão realizados. Em seguida, foi inserido o endereço de rede de destino, nesse caso o site da prefeitura de Brumadinho. Também foi adicionado o comando *"|tee"*, seguido de um "nome", para geração do arquivo de *log* com os registros dos disparos realizados, conforme mostrado abaixo:

ping -c 95000 brumadinho.mg.gov.br | tee ping_brumadinho.log

Também foi realizado um mapeamento dos saltos necessários para que o pacote *ICMP* trafegasse de Uberlândia até a prefeitura de Brumadinho, utilizando o comando *traceroute*. O comando *traceroute* tenta rastrear a rota que um pacote *IP (Internet Protocol)* segue para um destino na Internet, ativando os pacotes de análise *UDP (User Datagram Protocol)*.

Após iniciado, o comando *traceroute* começa a rastrear os saltos que um pacote *IP* realiza até atingir um determinado destino na rede, ativando, assim, pacotes de análise UDP com um tempo máximo de vida que se encerra no primeiro dispositivo que o receber. Em seguida, atendendo a uma resposta *ICMP TIME_EXCEEDED*, de *gateways* ao longo do caminho, as análises são iniciadas com um valor de um salto, que é acrescido em um salto por vez até que uma mensagem *ICMP PORT_UNREACHABLE* seja retornada. Essa mensagem de retorno indica que o destino foi localizado ou que o comando atingiu o número máximo de saltos permitidos para o rastreio. Dessa forma, o comando vai mapeamento cada destino atingido, e vai mostrando em tela cada ativo na Internet por qual recebe um retorno até que tenha percorrido toda a rota entre dois dispositivos quaisquer interligados por enlaces de dados. Nesse caso, a rota entre a máquina virtual na central em Uberlândia e a prefeitura de Brumadinho.

O comando *traceroute* envia três análises em cada salto para registrar os seguintes parâmetros:

- ❏ Valor de saltos
- ❏ Endereço do *gateway*
- ❏ Tempo de ida e volta de cada análise bem-sucedida

Quando o *traceroute* recebe as respostas da análise, caso essas venham de *gateways* diferentes, o comando imprimirá o endereço de cada sistema que estiver respondendo. Se não houver resposta de uma análise, em um intervalo de tempo limite de três segundos, um "*" (asterisco) será impresso para essa análise (TRACEROUTE, 2022).

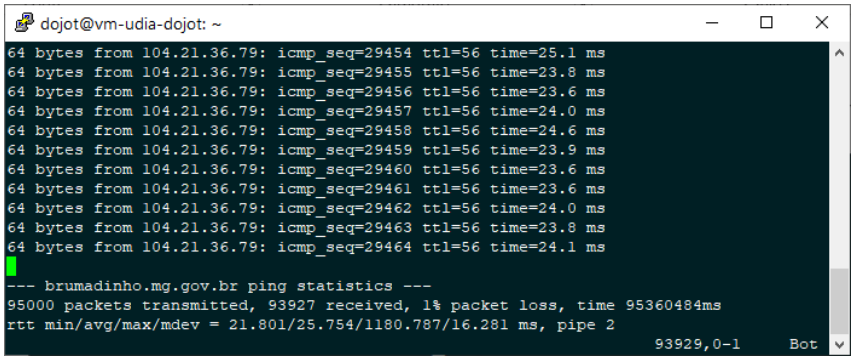

Figura 14 – Resultado comando *PING* de Uberlândia para Brumadinho. (Fonte: Do autor (2022)).

Para realizar o mapeamento dos saltos necessários entre a máquina virtual na central em Uberlândia até a prefeitura de Brumadinho, foi utilizado o comando *traceroute*, seguido do endereço do site da prefeitura de Brumadinho, conforme mostrado abaixo.

***traceroute* brumadinho.mg.gov.br**

Os resultados desse experimento estão descrito na Seção 5.1.2.

5.1.2 Resultados

Como saída ao comando de *PING*, disparado da máquina virtual na central em Uberlândia para a prefeitura de Brumadinho, são obtidas as respostas do protocolo *ICMP*. Conforme mostrado na Figura 14, o tamanho do pacote é de 64 bytes, e o endereço de destino, nesse caso a prefeitura de Brumadinho, é 104.21.36.79 no momento em que o experimento foi realizado. Também é mostrado o número de sequência *ICMP* para controle dos pacotes enviados e recebidos, e o *TTL Time to live*. Já o *Time*, que é dado em milissegundos, representa o intervalo de tempo gasto na comunicação.

O comando *PING*, ao final de sua execução, também mostra algumas estimativas gerais obtidas na saída. Conforme mostrado na Figura 14, foram disparados 95000 pacotes *ICMP*, de acordo com o que foi configurado previamente, porém somente 93927 foram recebidos, o que totaliza cerca de 1% de perda de pacotes.

De posse dos valores obtidos no campo *TIME*, que representam a latência na comunicação entre Uberlândia e Brumadinho, foram calculados os valores de mínimo, máximo, média aritmética e desvio padrão respectivamente.

- ❏ Mínimo: Pacote recebido no menor intervalo de tempo, em milissegundos.
- ❏ Máximo: Pacote recebido com o maior intervalo de tempo gasto, em milissegundos.
- ❏ Média: Média aritmética de todas as amostras coletadas, em milissegundos, divido pelo número de amostras.

Tabela 5 – Resultado dos cálculos experimento 24 horas Brumadinho.

Brumadinho Conjunto 24 horas			
Máximo	Mínimo	Média	Desvio
99.6	10.8	18.09838	7.82095

Tabela 6 – Resultado dos cálculos experimento com fração de 4096 amostras Brumadinho

Brumadinho Distribuição 4096			
Máximo	Mínimo	Média	Desvio
95.9	11.6	16.7855	7.52133

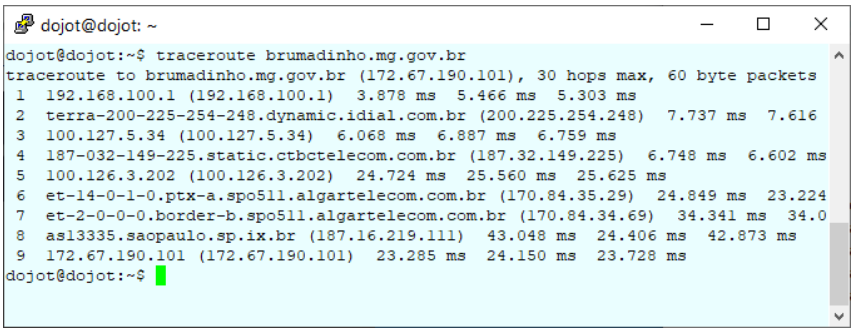

Figura 15 – Resultado Traceroute de Uberlândia para Brumadinho.(Fonte: Do autor (2022)).

- Desvio Padrão: Medida que expressa o grau de dispersão do conjunto de dados, ou seja, indica o quanto um conjunto de dados é uniforme. Quanto mais próximo de 0 for o desvio padrão, mais homogêneo são os dados.

As Tabelas 5 e 6 sumarizam os resultados do experimento 5.1.1. É possível notar que os valores de média, máximo, mínimo e desvio padrão estão muito próximos entre as duas tabelas, o que indica a equivalência entre eles. Esta similaridade garante que arquivo de 4096 amostras reproduz os valores de atraso na comunicação.

Como o *TC* utiliza por padrão arquivos de distribuição com 4096 valores, foram selecionadas aleatoriamente 4096 amostras do conjunto de dados obtidos no experimento de 24 horas. Em seguida, os mesmos valores mínimo, máximo, média e desvio padrão foram calculados, a fim de verificar a semelhanças existentes no arquivo gerado.

Portanto, é possível concluir que o arquivo de distribuição com 4096 amostras é equivalente ao conjunto completo de dados, uma vez que os valores mínimo, máximo, média e desvio padrão estão muito próximos uns dos outros. Dessa forma, o *TC* irá reproduzir a real variação de latência na comunicação dos enlaces de dados ao longo do tempo entre a máquina virtual de Uberlândia e a máquina virtual de Brumadinho. Desse modo, fica reproduzido, de forma confiável, os atrasos na comunicação entre as duas cidades, alcançando o objetivo descrito na Seção 1.2.

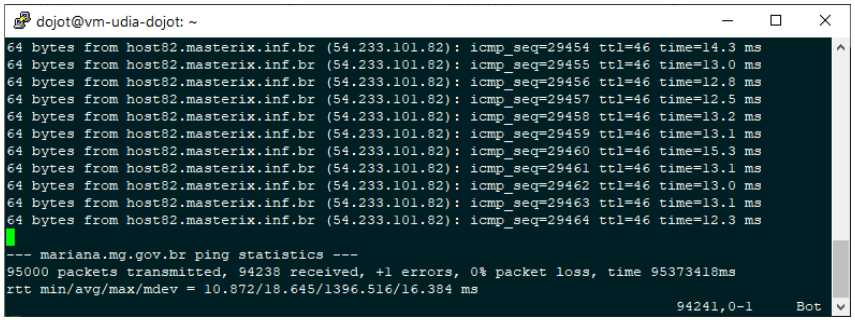

Figura 16 – Resultado comando PING de Uberlândia para Mariana.(Fonte: Do autor (2022)).

Como resultado, a execução do comando *traceroute*, conforme mostrado na Figura 15, apresenta um total de nove saltos, quando se dispara um pacote da máquina virtual, que está em Uberlândia até alcançar a prefeitura de Brumadinho. Esses dados são cruciais para entender melhor como se dá a comunicação entre os enlaces de dados das duas cidades. A título de exemplo, caso a prefeitura de Brumadinho tivesse seu site hospedado em uma solução de computação na borda ou névoa, que são soluções que operam próximo ao cliente, o número de saltos seria muito menor, podendo em alguns casos, até ser fisicamente dentro da rede da própria provedora de Internet. Nesse exemplo citado, não estaria representado o real cenário de latência dos enlaces de dados entre as duas cidades.

De acordo com a Figura 15, é possível notar que ocorreram nove saltos de Uberlândia até a prefeitura de Brumadinho. Portanto, é possível concluir que não existe nenhuma solução de computação de borda ou de névoa implementada, e que os pacotes estão saindo de fato da cidade de Uberlândia, como mostrado no salto 5, onde a latência passa para 24ms, sendo este salto quadro vezes maior do que no salto 4. Dessa forma, os nove saltos demonstram que o experimento para aferir a latência entre as duas cidades foi bem sucedido e reproduz o cenário real de forma fidedigna.

5.2 Aferindo latência entre Uberlândia e Mariana

Como saída ao comando de *PING*, disparado da máquina virtual na central em Uberlândia para a prefeitura de Mariana, são obtidas as respostas do protocolo ICMP. Conforme mostrado na Figura 16, o tamanho do pacote é de 64 bytes. O endereço de destino, nesse caso a prefeitura de Mariana, é 54.233.101.82 no momento em que o experimento foi realizado. Também são mostrados os números de sequência *ICMP*, o *TTL* e o *Time*, que é dado em milissegundos, representando o intervalo de tempo gasto na comunicação.

O comando *PING*, ao final de sua execução, também mostra as estimativas gerais obtidas na saída. Conforme mostrado na Figura 16, foram disparados 95000 pacotes ICMP, configurados previamente, porém somente 94238 foram recebidos, o que totaliza cerca de menos de 1% de perda de pacotes.

Tabela 7 – Resultado dos cálculos experimento 24 horas Mariana.

Mariana 24h			
Máximo	Mínimo	Média	Desvio Padrão
99,6	10,8	17,51846986	7,850452167

Tabela 8 – Resultado dos cálculos experimento com fração de 4096 amostras Mariana.

Mariana Distribuição 4096			
Máximo	Mínimo	Média	Desvio Padrão
95,7	11,3	17,3357666	7,161139674

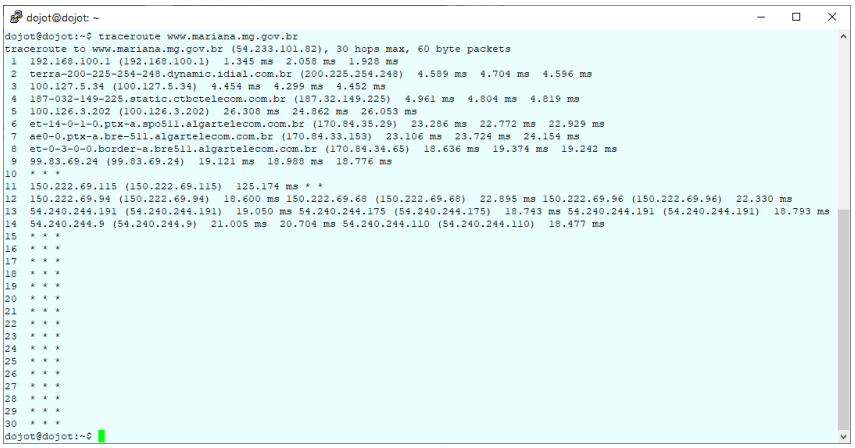

Figura 17 – Rastreamento Traceroute de Uberlândia para Mariana.(Fonte: Do autor (2022)).

De posse dos valores obtidos no campo *TIME*, que representa a latência na comunicação entre os enlaces de dados de Uberlândia e Mariana, foram calculados os valores de mínimo, máximo, média aritmética e desvio padrão, respectivamente, conforme mostrado na Tabela 7.

Como o *TC* utiliza, por padrão, arquivos de distribuição com 4096 valores foram capturadas aleatoriamente 4096 amostras, do conjunto de dados obtidos no experimento de 24 horas. Em seguida, os mesmos valores de mínimo, máximo, média e desvio padrão foram calculados, a fim de verificar a precisão do arquivo gerado, conforme mostrado na tabel- 8.

Logo, é possível concluir que o arquivo de distribuição com 4096 amostras é equivalente ao conjunto completo de dados, uma vez que os valores mínimo, máximo, média e desvio padrão estão muito próximos uns dos outros. Dessa forma, o *TC* irá reproduzir a real variação de latência dos enlaces de dados ao longo do tempo entre a máquina virtual de Uberlândia e a máquina virtual de Mariana. Desse modo, fica reproduzido, de forma confiável, os atrasos na comunicação entre as duas cidades, alcançando o objetivo descrito na Seção 1.2.

Conforme mostrado na Figura 17, a execução do comando *traceroute* tem um total de

trinta saltos quando se dispara um pacote da máquina virtual que está em Uberlândia até alcançar a prefeitura de Mariana. Esses dados são muito importantes para entender melhor como se dá a comunicação entre as duas cidades.

Com o auxílio da Figura 17, é possível concluir que não existe nenhuma solução de computação de borda ou de névoa implementada no site da prefeitura de Mariana. Como mostrado no salto 5, onde a latência passa para 26ms, e novamente no salto 11, onde o atraso sobe para 125ms, esses aumentos indicam propagação por meio físico, ou seja, distância geográfica. Dessa forma, os trinta saltos demonstram que o experimento para aferir a latência entre os enlaces de dados das duas cidades foi bem sucedido.

Cabe ressaltar que, diferentemente do experimento de Brumadinho, onde todos os nove saltos obtiveram resposta da análise UDP com endereço de *IP* ou nome de domínio, no experimento de Mariana metade dos saltos não obteve resposta. No experimento de Mariana os saltos de 15 a 30, marcados com "asterisco", significam que o rastreamento não obteve resposta no que diz respeito ao endereço de *IP* e ao nome de domínio do ativo de rede que realizou a tratativa do pacote. Isso ocorre, muitas vezes, por questões de segurança, já que endereços conhecidos podem ser alvos de ataques na Internet.

5.3 Transferência de aprendizado em cenários de desastres

Com base na infraestrutura virtualizada que foi construída, conforme descrito no Capítulo 4 e configurado com base no Experimento 5.1.1, foram trafegados dados pelos enlaces de comunicação que representam as cidades de Uberlândia, Brumadinho e Mariana. No entanto, em uma rede de dispositivos de sensoriamento *IoT* virtualizados, os dados gerados são todos sintéticos, o que não era ideal mediante a proposta da dissertação. Para resolver essa questão, foram utilizados dados reais, coletados em ambiente real obtidos em (PASQUINI; STADLER, 2017). Esses dados foram utilizados nos treinamentos de aprendizado de máquina com o algoritmo *Random Forest Regressor*, uma vez que as predições buscam rótulos numéricos, para cada grupo de dados que está proposto analisar, respectivamente.

5.3.1 Visão Geral

Este experimento consiste em realizar predições quanto a cenários de desastres utilizando as informações que foram trafegadas pela infraestrutura construída, estas predições foram realizadas primeiramente na central de monitoramento utilizando a coleta de dados a cada um segundo, que é aqui chamada de modelo padrão. Essas previsões realizadas pela implementação do algoritmo *Random Forest Regression* foram avaliadas utilizando o *NMAE* como parâmetro de comparação uma vez que se trata de um regressor .

De forma similar ao que foi feito para o modelo padrão, o modelo foi treinando utilizando todos os tempos de leitura da proposta da dissertação respectivamente: 1, 30, 60, 120 e 300 segundos. Também foi gerado o *NMAE* de cada um, afim de realizar comparações entre eles.

Em um segundo momento foi implementado a transferência de aprendizado, onde foi realizado treinamento do algoritmo de aprendizado de máquina nos locais de monitoramento, utilizando as leituras de dados a cada 300 segundos, e este modelo treinado foi repassado a central. O envio dos dados a cada 300 segundos aqui representa o maior intervalo de tempo dentre o que está sendo proposto, e também o menor consumo de recursos da infraestrutura para realizar o tráfego dos dados. Por fim também foi realizado o treinamento do algoritmo para os demais tempos de leitura propostos na dissertação: 1, 30, 60 e 120 segundos, e enviados a central.

O objetivo destes experimentos é verificar se a transferência de aprendizado pode ser aplicada como uma ferramenta habilitadora para o monitoramento a cenários de desastre em regiões afastadas.

Para o desenvolvimento do algoritmo de aprendizado de máquina, *Random Forest Regressor*, sobre o conjunto de dados obtido em (PASQUINI; STADLER, 2017), foi utilizada a linguagem *Pyhton*, que é mantida pelo *Pyhton Software Foundantion* (Fundação do software *Pyhton* em português) (PYTHON, 2022).

A linguagem utilizada foi *Pyhton*, versão 3 (PYTHON, 2022), e para implementação do código, foi utilizado o notebook *Jupyter*, que é um software de código aberto e está disponível no GitHub (SOFTWARE, 2022).

No desenvolvimento do algoritmo de aprendizado de máquina, *Random Forest Regressor*, primeiramente são adicionadas as bibliotecas necessárias para utilização do algoritmo. Também são adicionadas algumas bibliotecas para geração dos gráficos. A biblioteca *sklearn* (SCIKIT, 2022) foi utilizada para criação das árvores de decisão e para o algoritmo *Random Forest Regressor*.

Após a inserção das bibliotecas, utilizando os arquivos com os dados obtidos em (PASQUINI; STADLER, 2017), foi realizada uma divisão dos dados: o primeiro conjunto de dados utilizou uma leitura por segundo, que representa o modelo padrão. No processo de treinamento do algoritmo de aprendizado de máquina, os dados foram separados, sendo 30% dos dados para teste, e 70% para treino do algoritmo *Random Forest Regressor*. Os demais conjuntos de dados foram criados para cada treinamento respectivamente:

- **1 Segundo:** *X_train e y_train*, correspondente a 100% do conjunto de dados, com treinamento a cada um segundo.

- **30 Segundos:** *x_t30s e y_t30s* correspondente a um conjunto de treinamento com envio de informações a cada trinta segundos.

- **60 Segundos:** *x_t60s e y_t60s* correspondente a um conjunto de treinamento com envio de informações a cada sessenta segundos.
- **120 Segundos:** *x_t120s e y_t120s* correspondente a um conjunto de treinamento com envio de informações a cada cento e vinte segundos.
- **300 Segundos:** *x_t300s e y_t300s* correspondente a um conjunto de treinamento com envio de informações a cada trezentos segundos.

Após a criação dos conjuntos de dados, para cada um dos conjuntos propostos, foram criadas florestas de árvore de regressão.

Com os modelos criados, foi realizada a predição dos valores para o conjunto de dados de validação, respectivamente para cada um dos tempos de leitura, 1, 30, 60, 120 e 300 segundos respectivamente.

De posse dos valores de predição, para cada um dos conjuntos de dados de validação, foi calculado o *NMAE*, conforme mostrado abaixo, que é erro médio absoluto normalizado, utilizado para mensurar o quão assertivo o modelo de regressão foi.

$nmae1s\_test = (abs(pred1s\_test - y\_t300s\_test).mean())/y\_t300s\_test.mean()$

O *NMAE* é obtido utilizando a função "Abs"que retorna o valor absoluto da diferença entre a predição realizada pelo algoritmo, e o conjunto de teste, dividido pelo conjunto de teste.

Com os *NMAEs* calculados e armazenados em listas, uma nova lista foi criada, de modo a conter todos os conjuntos de listas gerados, nos cálculos de *NMAE*, para cada um dos tempos da proposta.

De modo a simular o ambiente de desastres para o cenário com maior intervalo dentre o que está sendo proposto, cujo os dados não são recebidos a cada segundo, mas sim a cada 300 segundos, foi realizada a predição dos valores para o conjunto de dados de teste. Aqui temos a transferência de aprendizado, conforme é possível visualizar a Figura 18, onde o modelo é treinado no local de monitoramento e então enviado a central. Treinar o modelo com dados coletados a cada segundo nos locais de monitoramento é factível já que os dados não precisam ser movimentados pela infraestrutura. Nessa etapa o modelo treinado a cada segundo foi envidado para a central juntamente com uma fração de dados coletados a cada 300 segundos.

De modo a simular o ambiente de desastres para todos os cenários possíveis que estão sendo propostos na dissertação, cujo dados não são recebidos a cada segundo, mas sim a cada 30, 60, 120 e 300 segundos, foi realizada a predição dos valores para o conjunto de dados de teste.

De posse dos valores de predição para cada um dos conjuntos de dados de teste de todo os cenários que estão sendo propostos, foi calculado o *NMAE*.

Da mesma maneira para o conjunto de dados de validação, também foi calculado aqui o *NMAE* para o conjunto de dados de testes. De acordo com a proposta da dissertação,

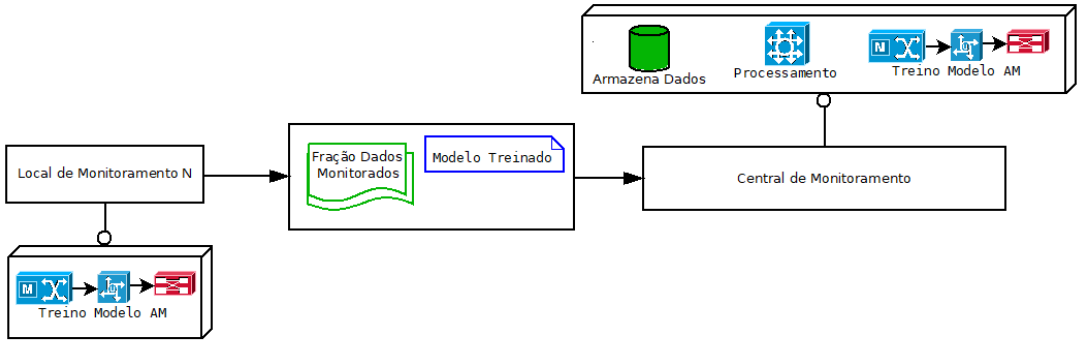

Figura 18 – Transferência de aprendizado. (Fonte: Do autor (2022)).

as predições realizadas utilizaram os dados recebidos apenas a cada 300 segundos, o que representa o maior intervalo de tempo dentre o que esta sendo proposto.

Similar ao conjunto de dados de validação, as listas criadas para o conjunto de testes também foram agrupadas em uma lista com os *NMAEs* calculados. Essas listas foram utilizadas para armazenar os resultados dos cálculos em cada rodada de execução do código. Essa descrição detalhada representa uma rodada de repetição do código, afim de elevar a precisão do modelo de aprendizado de máquina, cujo código foi executado por 3000 vezes. Primeiramente para o conjunto de dados de "leitura", e depois para o conjunto de dados de "escrita".

Após o algoritmo ter executado por três mil vezes, foram gerados gráficos para melhor representar os dados e as discrepâncias existentes entre o modelo padrão e a proposta da dissertação. A geração dos gráficos foi desenvolvida no código *python* e esta disponível no *GitHub* (COLANTONI, 2022).

Os dados, utilizados no experimento de aprendizado de máquina, foram obtidos por meio do *Github* (PASQUINI; STADLER, 2017), *Learning From Network Device Statistics*.

As estatísticas coletadas no trabalho em questão foram armazenadas em arquivos "CSV" (valores separados por vírgulas), que é um arquivo de texto, com formato específico, que possibilita salvar os dados em um formato estruturado de tabela. Esses arquivos contêm "m" linhas, de "n" recursos. Cada linha representa uma observação e tem um *TimeStamp*, indicando quando as estatísticas foram medidas. As métricas de nível de serviço coletadas foram armazenadas em arquivos "Y.csv" junto com o tempo de observação. Durante os experimentos, as estatísticas "X" e "Y" são coletadas a cada segundo no banco de testes.

A primeira linha de todos os arquivos CSV traz os rótulos para os "N" recursos em "X" e para as métricas de nível de serviço "Y". No arquivo "X.csv", o rótulo de um determinado recurso "x" é representado, usando a estrutura *"Index_FeatureName"*. Conforme mencionado em (PASQUINI; STADLER, 2017), esse conjunto de dados possui 48 recursos, pois o caminho possui 12 interfaces.

Os dados contidos nos dois arquivos CSV foram utilizados para realizar os experi-

mentos de aprendizado de máquina, e também foram trafegados dentro da infraestrutura virtualizada, construída nesta dissertação.

5.3.2 Conjunto de dados

O conjunto de dados trafegado pela infraestrutura desenvolvida foi obtido em(PASQUINI; STADLER, 2017). O referido conjunto consiste em um aglomerado de métricas obtidas no monitoramento fim-a-fim de uma infraestrutura física, que é formada de um conjunto de servidores em uma rede *OpenFlow*. Os dados foram originalmente gerados monitorando a infraestrutura enquanto um serviço de vídeos sobre demanda foi executado. Os autores buscavam realizar estimativas de métricas de infraestrutura com *QoS* de ponta a ponta entre o conjunto de servidores e os clientes do serviço.

No trabalho Pasquini e Stadler (2017) foram coletadas cerca de quarenta e oito métricas da infraestrutura, como uso da *CPU*, memoria *RAM*, rede, entre outros, sendo cada entrada registrada com um *TimeStamp*. Para cada métrica armazenada em um arquivo "X.csv", também foi gerado o respectivo registro de "leitura" e "escrita" com o *TimeStamp* correspondente, em um arquivo "Y.csv". O autores implementaram um algoritmo de regressão, e para verificar a eficácia das predições realizadas foi calculado o *NMAE*.

5.3.3 Treinamento teste e avaliação do modelo

O treinamento do modelo de inteligência artificial, consistiu primeiramente da construção do modelo padrão, onde a leitura é feita a cada segundo, de posse destes dados foram realizados testes e calculado o *NMAE*. Com o modelo padrão definido, foi possível estabelecer um parâmetro de comparação para os tempos de leitura propostos na dissertação respectivamente: 1, 30, 60, 120 e 300 segundos. Também foi gerado o *NMAE* de cada um, afim de realizar a confrontação entre eles.

Os tempos de leitura da proposta representam dados parciais, uma vez que foi investigado o uso da transferência de aprendizado como ferramenta habilitadora a monitoramento em cenários de desastre.

O *NMAE* foi escolhido como parâmetro para comparação entre os modelos por se tratar de um problema de regressão onde é possível determinar o grau em que as variáveis independentes influenciam as variáveis dependentes, conforme descrito na Seção 2.4.

O *NMAE* também permite mensurar se o modelo treinado realizou previsões de forma satisfatória, e como foi aplicado nesses experimentos tanto para o modelo padrão como para todos os tempos de leitura propostos, se torna o parâmetro de comparação, conforme foi inicialmente planejado na Seção 1.2.

Tabela 9 – Experimento 5.3.4: Todos os modelos com menor conjuntos de dados.

Modelos	Abordagem Tradicional	Proposta da Dissertação
1s	1s	300s
30s	30s	300s
60s	60s	300s
120s	120s	300s
300s	300s	300s

5.3.4 Resultados

Com os modelos treinados, os *NMAEs* calculados, tanto para o conjunto de dados de "leitura" quanto para o conjunto de dados de "escrita", foram gerados gráficos, a fim de melhor representar os resultados obtidos nos experimentos realizados.

Conforme é possível visualizar na Tabela 9, a abordagem tradicional utiliza para cada modelo treinado o seu respectivo conjunto de dados, sendo o modelo treinado a cada 1 segundo com o conjunto de dados de 1 segundo, o modelo treinado a cada 30 segundos com o conjunto de dados de 30 segundos, e assim por diante ate o modelo treinado a 300 segundos. A proposta da dissertação utilizou o mesmo conjunto de dados, o de 300 segundos que é o menor dentre os propostos, para todos os modelos treinados. Dessa maneira todos os modelos são avaliados utilizando o conjunto que menos consome recursos dos enlaces de dados.

Conforme mostrado na Figura 19, onde temos um gráfico de modelos *Random Forest* por *NMAEs* calculados, é possível observar duas métricas, uma sendo a abordagem padrão, que presenta uma leitura convencional dos dados obtidos em campo e transmitidos em sua totalidade para a unidade central, e a outra é a proposta da dissertação para o dado de "leitura" do arquivo Y.csv. O mesmo é feito para dado de "escrita" conforme é possível visualizar na Figura 20.

Com esses resultados apresentados, é possível verificar que ambas as séries contidas nos gráficos apresentam uma pequena variação do *NMAE*. E, para os casos de amostras a cada 30 segundos e de amostras a cada 60 segundos, o *NMAE* chegou a ser menor do que para o modelo padrão.

Vale ressaltar ainda que a proposta consegue ser melhor no modelo de 1 segundo, na posição (0.0), ou seja, se treinarmos o modelo a cada 1 segundo em Brumadinho ou em Mariana e enviarmos o modelo para Uberlândia, realizando, assim a transferência de aprendizado, poderemos enviar dados a cada 300 segundos para a central, e o modelo terá praticamente a mesma *NMAE*.

Aqui ficam comprovadas as hipóteses descritas na Seção 1.3, uma vez que foi demonstrada a economia de dados sendo trafegados pela rede. A transferência de aprendizado se mostra habilitadora do monitoramento em região remota, já que não é preciso uma alta largura de banda entre os enlaces de dados. Dessa maneira, os dados de leituras podem ser enviados a cada 300 segundos, mantendo o *NMAE* muito próximo da abordagem

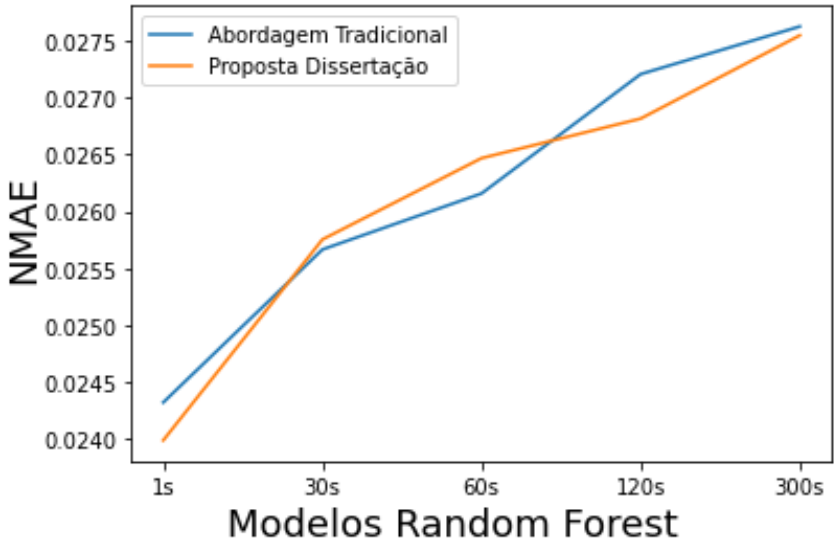

Figura 19 – Leitura - Modelos *Random Forest* por *NMAE* com dados a cada 300 segundos.(Fonte: Do autor (2022)).

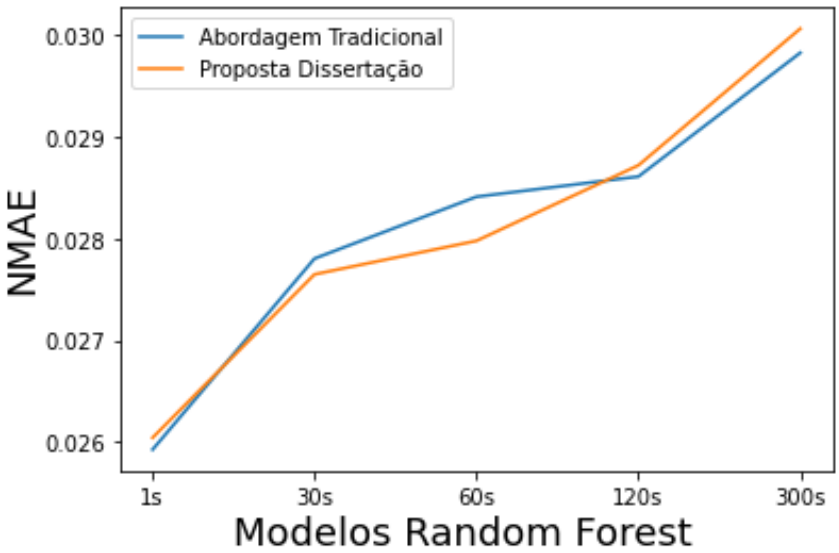

Figura 20 – Escrita - Modelos *Random Forest* por *NMAE* com dados a cada 300 segundos.(Fonte: Do autor (2022)).

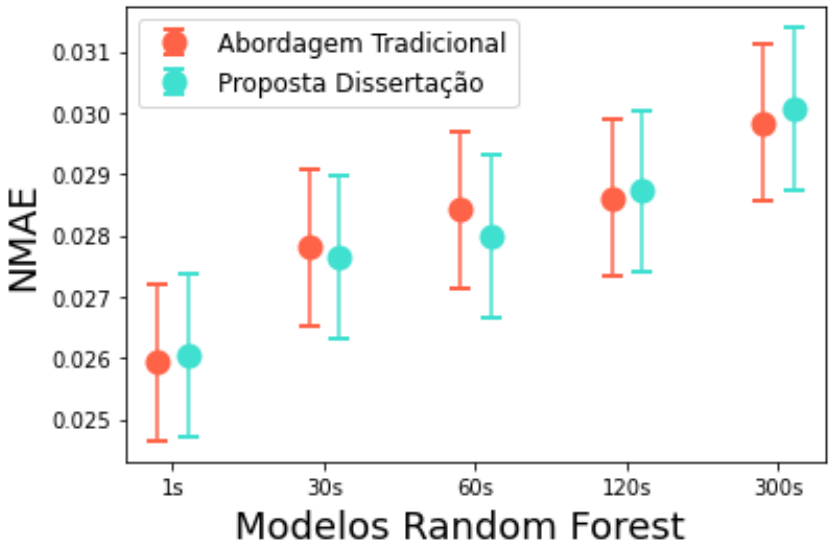

Figura 21 – Escrita - Desvio padrão com dados a cada 300 segundos.(Fonte: Do autor (2022)).

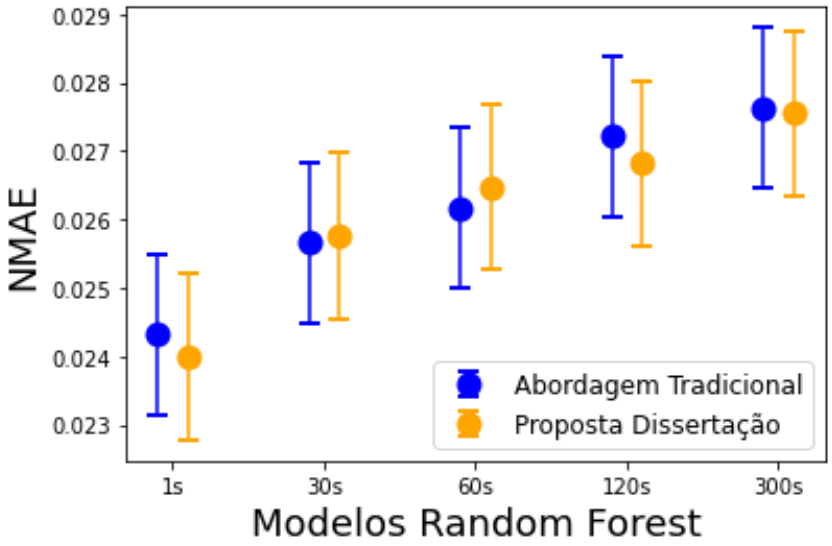

Figura 22 – Leitura - Desvio padrão com dados a cada 300 segundos.(Fonte: Do autor (2022)).

tradicional, onde é feita a cada 1 segundo.

Na busca por discrepâncias entre a abordagem tradicional, e a proposta da dissertação, foram calculados os desvios padrões. O cálculo foi realizado para todos os modelos de leitura propostos com dados enviados a cada 300 segundos, conforme mostra a Figura 21 para os modelos treinados para escrita, e a Figura 22 para os modelos treinados com os dados de leitura.

Analisando os gráficos das Figuras 21 e 22 é possível visualizar que ao treinar o modelo nos locais de monitoramento com dados a cada segundo e enviar o modelo treinado para

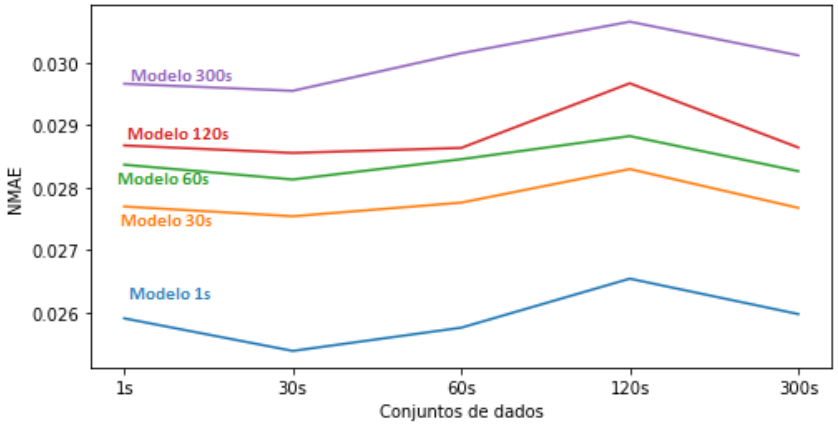

Figura 23 – Leitura - *NMAE* por conjuntos de teste.(Fonte: Do autor (2022)).

central, juntamente com uma fração dos dados lidos as cada 300 segundos, é obtido a mesma acurácia. Desta forma o monitoramento se torna factível já que não é preciso movimentar fluxos massivos de dados pela infraestrutura, validando a hipótese levantada na Seção 1.3.

Ambos os gráficos de desvio padrão, Figura 21 e Figura 22, apresentam o desvio padrão que representa o grau de variação do conjunto de elementos de cada algoritmo que foi treinado. Em ambos os casos, de "leitura" e de "escrita", os desvios padrões se afastaram muito pouco, quando comparados a abordagem tradicional com a proposta da dissertação. Pode-se, assim, concluir que a precisão da proposta, que seria a variação do erro *NMAE*, é praticamente a mesma da abordagem tradicional, uma vez que a proposta trabalha apenas com uma fração dos dados coletados. Dessa maneira, ficam validadas as hipóteses descritas na Seção 1.3.

Conforme mostrado na Figura 23, onde se apresenta um gráfico de *NMAEs* calculados por conjuntos de dados, é possível observar a comparação entre todos os modelos treinados que foram transferidos a central de monitoramento e receberam os conjuntos de dados 1,30, 60, 120 e 300 respectivamente.

Na Figura 24, é mostrado um gráfico de *NMAEs* calculados por conjuntos de teste, com os conjuntos de dados 1, 30, 60, 120 e 300, são visualizados cinco series , representando cada uma um modelo treinado, para o dado de escrita.

Conforme é possível visualizar na Tabela 10, para todos os modelos treinados nos locais de monitoramento, foram realizadas predições utilizando todos os conjuntos de dados da dissertação. Para o modelo treinado a cada 1 segundo, foram realizadas predições utilizando os conjuntos de dados 1, 30, 60, 120 e 300 segundos, e dessa forma respectivamente para todos os modelos treinados.

Foram calculados os desvios padrões para os dados de "escrita" e de "leitura conforme mostra a Figura 25 e a Figura 26 respectivamente, afim de melhor demonstrar o grau de dispersão destes dados.

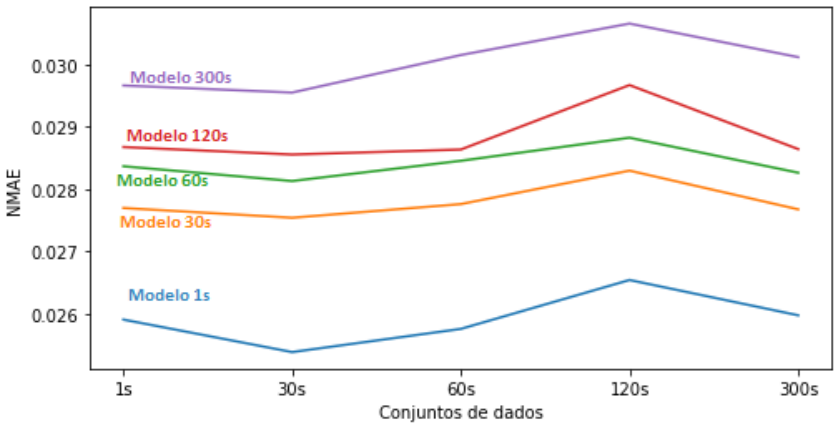

Figura 24 – Escrita - *NMAE* por conjuntos de teste.(Fonte: Do autor (2022)).

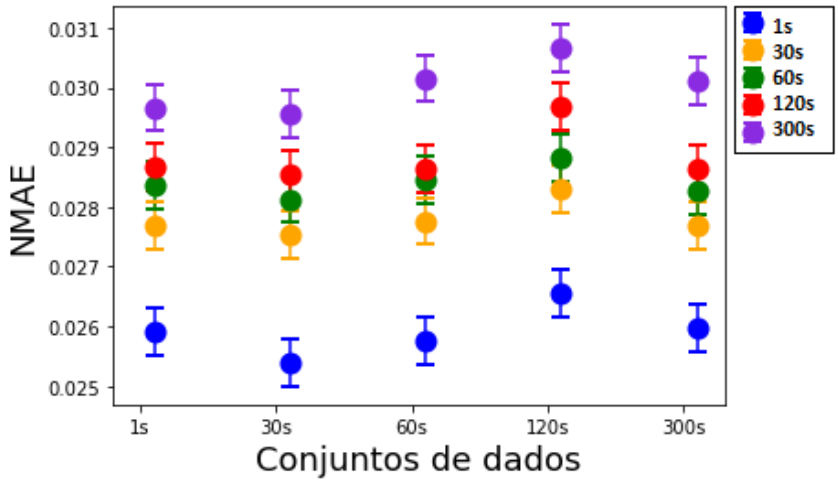

Figura 25 – Escrita - Desvio padrão.(Fonte: Do autor (2022)).

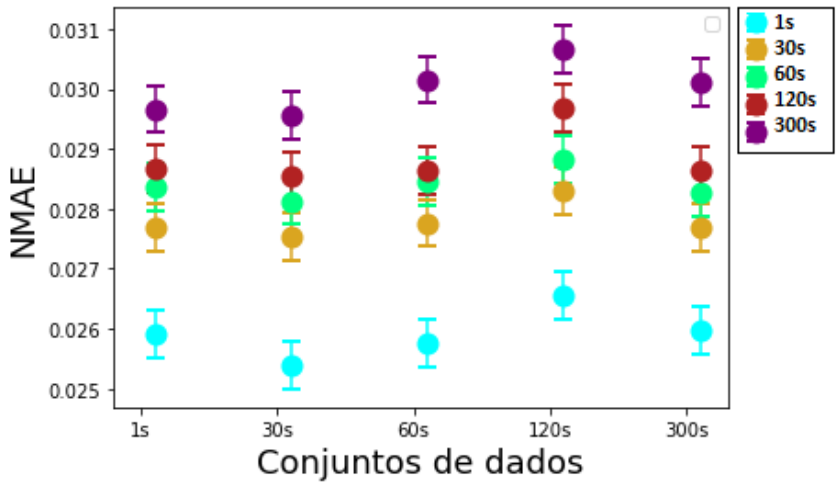

Figura 26 – Leitura - Desvio padrão.(Fonte: Do autor (2022)).

Tabela 10 – Experimento 5.3.4: Todos os modelos com todos os conjuntos de dados.

Modelos	Conjuntos de dados
1s	1s
	30s
	60s
	120s
	300s
30s	1s
	30s
	60s
	120s
	300s
60s	1s
	30s
	60s
	120s
	300s
120s	1s
	30s
	60s
	120s
	300s
300s	1s
	30s
	60s
	120s
	300s

Os resultados mostram que a transferência de aprendizado do modelo treinado nos locais de monitoramento enviado à central é viável. Com poucos dados, é possível saber precisamente, ou seja, com o mesmo nível de erro NMAE, o que está acontecendo na localidade monitorada, quando utilizamos o melhor modelo treinado a cada 1 segundo em conjunto com os dados enviados a cada 300 segundos.

5.4 Trafegando os dados pela infraestrutura

Utilizando a infraestrutura virtualizada que foi construída, conforme descrito no Capítulo 4, e os dados obtidos em (PASQUINI; STADLER, 2017), foi realizado o tráfego dos dados pela infraestrutura. Esse tráfego foi monitorado nas máquinas virtuais que representam as cidades de Uberlândia, Brumadinho e Mariana, respectivamente.

O monitoramento da infraestrutura, realizado durante o tráfego dos dados, foi utilizado para mensurar duas métricas, o volume de bytes utilizado na transmissão, e o número de pacotes gerados no tráfego dos dados pelos enlaces de dados que interligam os locais de monitoramento à central.

Para a leitura do trafego de rede, durante os disparos das mensagens *MQTT* dos locais de monitoramento para a central, foi utilizado o software *TCPdump*. O programa é de código aberto e, durante sua execução, imprime em tela uma descrição do conteúdo dos pacotes em uma interface de rede selecionada, e essa descrição é precedida por um carimbo de hora. Esse carimbo é impresso com horas, minutos, segundos e frações de segundo.

O *TCPdump* também pode ser executado com o parâmetro "-w", que faz com que ele salve os dados registrados em um arquivo para análise posterior. Também pode ser utilizado o parâmetro "-r", que faz com que o *TCPdump* leia de um arquivo de pacote salvo, em vez de ler de pacotes a partir de uma interface de rede. Outra forma de execução é com o parâmetro "-v", que faz com que ele leia uma lista de arquivos de pacotes salvos. Em todos os casos, apenas os pacotes, que correspondem à expressão inserida na linha de comando, serão processados pelo *TCPdump* (OPENSOURCE, 2022).

5.4.1 Volume de bytes e Número de Pacotes Transmitidos

Conforme a infraestrutura virtualizada que foi construída com uso de máquinas virtuais, descrita no Capitulo-4, tem-se um cenário onde cada cidade é representada por uma máquina virtual. Uberlândia é a central de monitoramento, Brumadinho e Mariana são os locais monitorados. Os dados obtidos em (PASQUINI; STADLER, 2017) são trafegados de Brumadinho e de Mariana para Uberlândia, de forma simultânea, utilizando o protocolo MQTT.

Para o envio dos dados, foi utilizado o software *Mosquitto_pub*, que é o cliente do *Mosquito*, utilizado para o envio das mensagens MQTT. Na linha de comando utilizada para disparo das mensagens MQTT, primeiramente é utilizado o comando, seguido do paramento "-h" para especificação do endereço de *IP* de destino. Nesse caso, o endereço do *broker* é o que está na instância do *dojot* em Uberlândia. Em seguida, o parâmetro "-p" especifica por qual a porta que se dará a comunicação. Por padrão do *Mosquito*, foi utilizada a porta 1883. Na sequência do comando, tem-se o parâmetro "-t", que é composto pelo usuário, que realiza a inserção da informação, e o ID do dispositivo que irá armazenar ou coletar os dados que estão sendo enviados. Por fim, tem-se o parâmetro "-m", cujas variáveis e os dados em si são inseridos, conforme mostrado no exemplo abaixo.

mosquitto_pub -h 10.10.100.134 -p 1883 -t admin70645f attrs -m '"temperatura":28,"umidade":17,"N":A'

Para o envio dos dados descritos no Capítulo 4, foi criado um *script* em *shell-script* do *Linux*, que realizou o disparo das mensagens MQTT.

O *script* criado é uma linguagem utilizada em ambiente de linha de comando, com extensão ".sh", que pode ser executado em computadores com *kernel Linux*, mais especificamente, nesse caso, os que fazem parte da distribuição Debian. Nas máquinas virtuais, foi utilizado o *Ubuntu server 20.1* como sistema operacional para a execução dos comandos:

mosquitto_pub -h 10.10.100.133 -p 1883 -t /admin/70645f/attrs
-m'{"temperatura":'$t',"umidade":'$u',"tempo":'$time',"data":"$data"}'
echo "Temperatura = "$t"Umidade = "$u"tempo="$d "data ="$LinhaCsv ;
sleep 1

Este *script* foi executado durante um período de trinta minutos para cada uma das propostas de monitoramento respectivamente. Na primeira rodada de execução, Brumadinho e Mariana executaram o *script* com o parâmetro "Sleep 1" durante trinta minutos, depois o mesmo *script* foi executado, modificando o parâmetro de espera respectivamente:

- **1 Segundo:** Tempo de execução de 30 minutos, com parâmetro "Sleep 1"
- **30 Segundos:** Tempo de execução de 30 minutos, com parâmetro "Sleep 30"
- **60 Segundos:** Tempo de execução de 30 minutos, com parâmetro "Sleep 60"
- **120 Segundos:**Tempo de execução de 30 minutos, com parâmetro "Sleep 120"
- **300 Segundos:** Tempo de execução de 30 minutos, com parâmetro "Sleep 300"

Dessa forma, todos os cenários propostos na dissertação foram ensaiados, simulando, assim, as possibilidades de cenários de desastre para todos esses casos. Durante o tráfego das informações, a rede foi monitorada de forma a estimar o volume de bytes e o número de pacotes trafegados durante os disparos das mensagens MQTT.

Para a leitura do tráfego de rede, foi utilizado o software *TCPdump*, que realiza o monitoramento em tempo real da interface de rede do computador ou do servidor que o está executando, e mostra em tela a leitura realizada. O comando utilizado com o parâmetro "-i" para especificar uma interface, no caso, foi configurado que qualquer pacote, sendo *TCP* na porta 1883, deveria ser capturado e armazenado em um arquivo de log respectivamente:

tcpdump -i any tcp port 1883 > log_1_udia.log
tcpdump -i any tcp port 1883 > log_1_brumadinho.log
tcpdump -i any tcp port 1883 > log_1_mariana.log
tcpdump -i any tcp port 1883 > log_30_udia.log
tcpdump -i any tcp port 1883 > log_30_brumadinho.log
tcpdump -i any tcp port 1883 > log_30_mariana.log
tcpdump -i any tcp port 1883 > log_60_udia.log
tcpdump -i any tcp port 1883 > log_60_brumadinho.log
tcpdump -i any tcp port 1883 > log_60_mariana.log
tcpdump -i any tcp port 1883 > log_120_udia.log
tcpdump -i any tcp port 1883 > log_120_brumadinho.log
tcpdump -i any tcp port 1883 > log_120_mariana.log

Tabela 11 – Resultado dos cálculos: Totais volume de bytes.

Volume de Bytes					
Cidades	**1S**	**30S**	**60S**	**120S**	**300S**
Uberlândia	1.634.448	44.969	17.386	8.693	5.441
Brumadinho	814.288	22.395	8.664	4.332	2.712
Mariana	820.164	22.084	8.261	4.130	2.729

Tabela 12 – Resultado dos cálculos: totais número de pacotes.

Número de Pacotes					
Cidades	**1S**	**30S**	**60S**	**120S**	**300S**
Uberlândia	17.963	492	191	96	60
Brumadinho	8.973	246	95	48	30
Mariana	8.990	241	91	46	30

tcpdump -i any tcp port 1883 > log_300_udia.log
tcpdump -i any tcp port 1883 > log_300_brumadinho.log
tcpdump -i any tcp port 1883 > log_300_mariana.log

5.4.2 Resultados

Após a execução de trinta minutos para cada uma das respectivas propostas de monitoramento, os *logs* foram gerados. De posso desses *logs*, foi calculado os totais de pacotes trafegados e o volume de bytes utilizados para o transporte dos dados entre os enlaces da infraestrutura.

A Tabela 11 mostra o volume de bytes que foi utilizado no transporte dos dados. As informações foram coletadas em uma das unidades de monitoramento até a central do dojot.

Na Tabela 12 , são mostrados os totais calculados a partir do número de pacotes que foram gerados durante a transmissão dos dados. Os valores foram calculados para as unidades de monitoramento, bem como os totais registrados na própria central.

Por fim, foram sumarizados cada uma das propostas de monitoramento, tanto para o volume de bytes quanto para o número de pacotes gastos no transporte dos dados.

Na Figura 27, é mostrado o volume de bytes para cada um dos modelos propostos. Como existe uma discrepância relevante, para o volume de bytes no cenário de monitoramento a cada 1 segundo, quando comparado aos demais cenários propostos, foi criado um outro gráfico com ampliação, mostrado na Figura 28. Esse gráfico ampliado mostra os cenários de 30, 60, 120, e 300 segundos, cujos cenários podem ser melhor visualizados.

Também foram gerados gráficos para melhor representar os resultados obtidos, nos totais calculados de número de pacotes, conforme mostrado na Figura 29. Neste gráfico, tem-se o número de pacotes pelos modelos propostos, existe uma discrepância similar ao volume de bytes, onde o cenário de monitoramento a cada 1 segundo é significativamente

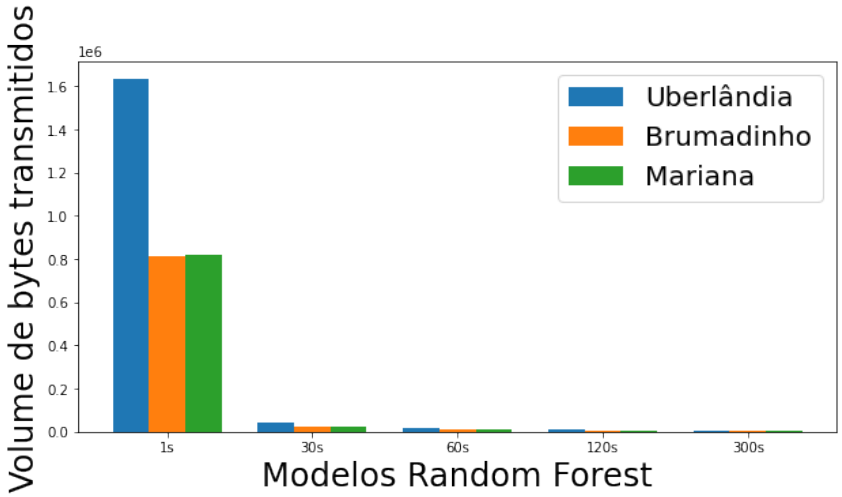

Figura 27 – **Tráfego de rede:** Volume de bytes.(Fonte: Do autor (2022)).

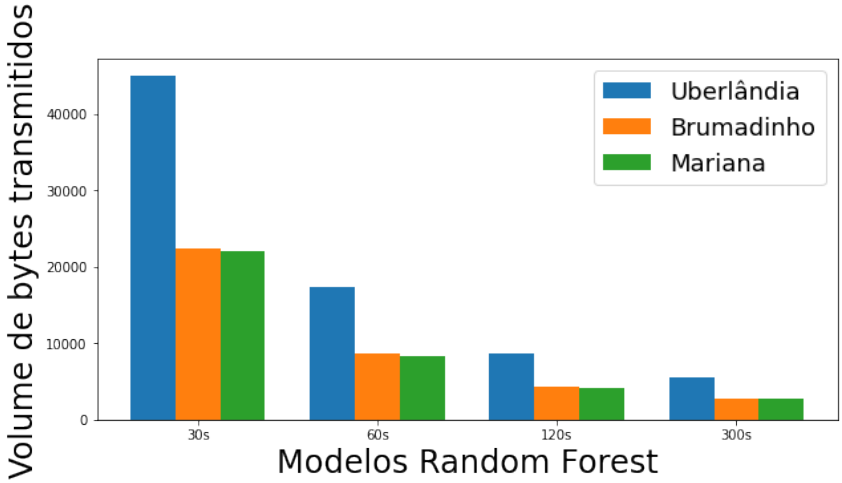

Figura 28 – **Tráfego de rede:** Volume de bytes Ampliação.(Fonte: Do autor (2022)).

maior do que os demais. Em virtude disso, foi criado um outro gráfico com ampliação, mostrado na Figura 30, dos cenários de 30, 60, 120, e 300 segundos, onde esses cenários podem ser melhor visualizados.

Analisando os gráficos, é possível concluir que, tanto o volume de bytes quanto o número de pacotes da central em Uberlândia, são próximos do dobro dos locais de monitoramento. Isso já era esperado, uma vez que a central do *dojot* está recebendo simultaneamente os dados dos dois locais de monitoramento. Dessa forma, é possível deduzir que a central sempre terá o volume de bytes e o número de pacotes próximo da soma dos N locais de monitoramento.

Os gráficos demostram que, expressivamente, os cenários propostos consomem volumes de bytes e números de pacotes significantemente menores, quando se compara a abordagem tradicional com a proposta da dissertação, cujos experimentos comprovam as hipóteses descritas na Seção 1.3. É notável que o cenário do pior caso, com envio dos dados a cada 300 segundos, consome menos recursos de infraestrutura de TIC no transporte dos dados

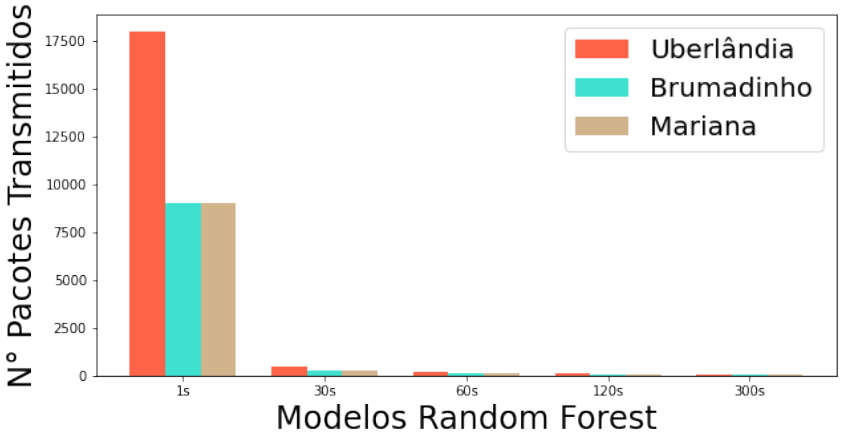

Figura 29 – **Tráfego de rede:** Número de pacotes.(Fonte: Do autor (2022)).

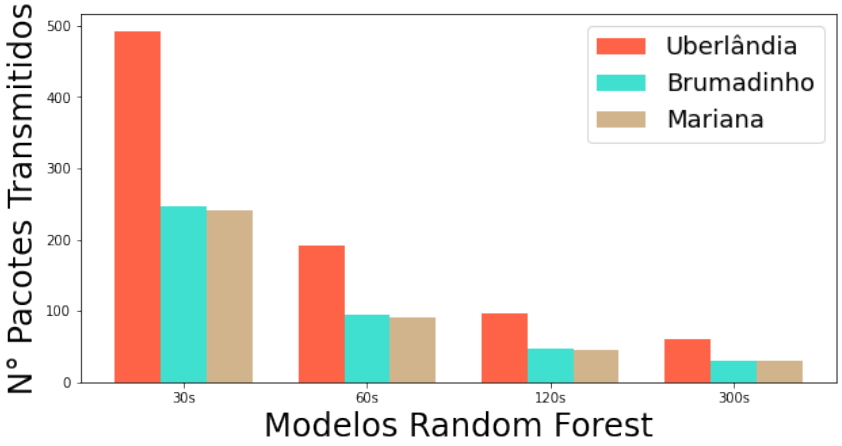

Figura 30 – **Tráfego de rede:** Número de pacotes Ampliação.(Fonte: Do autor (2022)).

entre os enlaces de dados dos locais de monitoramento. Assim sendo, considera-se que os objetivos descritos na Seção 1.2 foram alcançados.

Capítulo 6

Conclusão

A proposta da dissertação, que buscou investigar o uso da transferência de aprendizado como ferramenta habilitadora do monitoramento em zonas afastadas e locais carentes de infraestrutura de TIC, foi apresentada no Capitulo 4. As simulações realizadas para efetiva validação da proposta foi demonstrada no Capitulo 5, bem como os resultados dos experimentos, e os gráficos gerados. Nesse capítulo, foram detalhadas as contribuições que a proposta deste trabalho oferece à comunidade científica, como também as limitações do trabalho desenvolvido, e algumas sugestões de trabalhos futuros em continuidade ao que foi disposto.

Um dos maiores motivadores para o desenvolvimento da proposta foi a alta taxa de incidentes abordada no projeto *ADMITS*, como mostrado na Seção 1.1. Em especial, destacam-se os locais onde existem limitações quanto aos recursos de TIC, as falhas no funcionamento da Internet como é o caso das barragens localizadas nas cidades de Brumadinho e de Mariana, que sofreram desastres recentemente. Ao analisar o cenário atual dessas duas cidades, fica evidente a defasagem e a inviabilidade do modelo padrão de monitoramento, isso se deve às condições em que se encontram as cidades desse porte, pois não se vislumbram melhorias significativas nos recursos de infraestrutura de TIC.

Esta dissertação apresentou uma solução habilitadora do monitoramento nessas regiões para mitigação dos cenários de desastre, com uma instância do *dojot* como central de monitoramento, colaborando, dessa maneira, com o projeto *ADMITS*. Os locais onde a restrição na largura de banda e falhas na interoperabilidade da Internet são fatores limitantes quanto a sistemas de monitoramento, a proposta desenvolvida nesta dissertação se mostrou factível, uma vez que foi possível manter o monitoramento com um nível de erro *NMAE* muito próximo ao modelo padrão.

É importante ressaltar que os objetivos listados na Seção 1.2 foram cumpridos:

- ❏ Análise empírica da transferência de aprendizado como ferramenta habilitadora para monitoramento de locais remotos (Capítulo 5).
- ❏ Construção de uma arquitetura para ambiente *IoT* voltada para monitoramento de

desastres, com capacidade de emular distâncias geográficas e restrições de comunicação de dados (Capítulo 4).

- ❏ Implementação de algoritmos de aprendizado de máquina, capazes de realizar previsões a cenários de desastres (Seção 5.4).

6.1 Principais contribuições

O desenvolvimento deste trabalho de dissertação se iniciou com a identificação motivadora descrita na Seção 1.1 e as comparações estabelecidas no Capítulo 3 de trabalhos correlatos, que demonstraram que ainda não existia uma solução habilitadora com o cenário proposto na dissertação. A Seção 1.2, que trata dos objetivos, listou o foco do trabalho que foi descrito no Capítulo 4 e desenvolvido no Capítulo 5. Os objetivos foram alcançados conforme descritos abaixo.

- ❏ Contribuição ao projeto *ADMITS* com nova proposta (Capítulo 4).
- ❏ Solução habilitadora para monitoramento a cenários de desastre em regiões afastadas (Capítulo 5).
- ❏ Implementação de algoritmos de inteligência artificial para realização de predições de desastres (Seção 5.4).
- ❏ Construção da infraestrutura virtualizada (Seção 4.2), bem como toda a documentação detalhada para reprodução dessa infraestrutura (COLANTONI, 2022).
- ❏ Análise empírica da solução apresentada frente ao estado da arte (Seção 5.1.2, Seção 5.3.4, e Seção 5.4.2).

Dessa forma, foi possível inferir que a proposta da dissertação constitui uma contribuição inovadora. Toda a construção da infraestrutura virtualizada está detalhada no Capítulo 4, bem como o estudo estatístico realizado para configuração nos atrasos de comunicação, Seção 5.1.1 e Seção 5.2, entre as cidades monitoradas e a central. A descrição do desenvolvimento do modelo de inteligência artificial está descrito na Seção 5.4 e a geração dos gráficos para visualização da eficácia da proposta, quando comparada ao modelo padrão Seção 5.1.2, Seção 5.3.4, Seção 5.4.2. A apresentação da economia de recurso de infraestrutura de TIC, quanto ao volume de byte e ao número de pacotes necessários para o tráfego das informações entre os enlaces de dados, está demostrado nas Tabelas 11 e 12.

Embora todo o trabalho esteja permeado por conceitos do papel habilitador de monitoramento apresentado pela solução proposta a cenários de desastre, demonstrar que mesmo consumindo o que seria uma fração dos recursos necessários de infraestrutura de TIC para manter o monitoramento, com o *NMAE* equivalente ao modelo padrão, constitui o arcabouço desta dissertação.

Todo o código-fonte dos algoritmos de aprendizado de máquina desenvolvidos, bem como os comandos necessários para a construção da infraestrutura virtualizada com as respectivas imagens das máquinas virtuais, foram disponibilizados, e estão acessíveis publicamente por meio do *GitHub* em: (COLANTONI, 2022).

6.2 Limitações

O presente trabalho possui limitações quanto à degradação da infraestrutura, por isso não foram realizados testes com os experimentos, com perda de comunicação sendo superior a 300 segundos onde a fatia dos dados seria ainda menor. Também não foram realizadas implementações de outros algoritmos de aprendizado de máquina, além do *Random Forest Regression*, para verificar se haveria variação das predições realizadas e se essa modificação afetaria de forma significativa as variações de erro *NMAE* que foi o parâmetro utilizado para comparação da eficácia da proposta.

6.3 Trabalhos Futuros

Uma tecnologia que pode ser aplicada na contribuição ao projeto *ADMITS*, bem como a presente dissertação, é o *Federated Learning FL*. O aprendizado federado é uma técnica de aprendizado de máquina, que treina um algoritmo em vários dispositivos de borda descentralizados ou servidores que contêm amostras de dados locais, sem ocorrer a troca de informações entre os locais de monitoramento e a central. Essa abordagem contrasta com as técnicas tradicionais de aprendizado de máquina centralizado, em que todos os conjuntos de dados locais são carregados em um servidor. Também diverge das abordagens descentralizadas mais clássicas, que geralmente assumem que as amostras de dados locais são distribuídas de forma idêntica.

No aprendizado federado, é possível que vários ativos, que constituem a rede, construam um modelo de aprendizado de máquina comum e robusto, sem compartilhar dados. Esse modo de operação permite abordar questões críticas, como privacidade de dados, segurança de dados, direitos de acesso a dados e a dados heterogêneos.

Aplicar o aprendizado federado ao trabalho desenvolvido nesta dissertação poderia apresentar mais algumas soluções viáveis ao projeto *ADMITS*. Também haveria maior empregabilidade da solução, para setores onde a segurança dos dados é mais sensível. A falta de largura de banda na comunicação ou a carência em recursos de TIC de forma geral, uma vez que os dados não são compartilhados, seriam mitigados, o que tornaria sua adesão ainda mais aceitável na implementação de soluções em cenários de desastre.

Novamente, é de suma importância reiterar que a proposta da dissertação apresentou ótimos resultados nos experimentos demonstrados no Capítulo 5. E não foi descoberto qualquer impedimento à viabilidade da solução em zonas rurais ou em locais com falta

de recursos de infraestrutura de TIC. Assim sendo, a Seção 6.3 de Trabalhos Futuros, foi dedicada apenas à proposta de melhorias que podem ser realizadas na constante evolução do estado da arte.

Referências

AFONSO, A.; NUNES, C. **Probabilidades e Estatística. Aplicações e Soluções em SPSS. Versão revista e aumentada.** [S.l.]: Universidade de Évora, 2019.

AHMED, J. et al. Predicting sla conformance for cluster-based services using distributed analytics. In: IEEE. **NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium**. [S.l.], 2016. p. 848–852.

BRASIL, C. Cpqd brasil (2022). In: **CPQD Brasil (2022)**. [s.n.], 2022. Disponível em: <https://www.cpqd.com.br/>. Acesso em: 8.2.2022.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

BRUNEAU, P.; TAMISIER, T. Transfer learning and mixed input deep neural networks for estimating flood severity in news content. In: **MediaEval**. [S.l.: s.n.], 2019.

CANAVILHAS, J. A comunicação política na era da internet. In: LABCOM. **VIII Congresso Lusocom**. [S.l.], 2009.

COLANTONI, M. A. Git colantoni projeto admits dissertacao de mestrado. In: **GIT Colantoni ADMITS**. [s.n.], 2022. Disponível em: <https://github.com/mautkd/ADMITS>. Acesso em: 10.8.2022.

CUNHA, I. R. d. et al. Construção de mecanismo para suportar a predição de tempos de resposta do cassandra a partir de métricas de infraestrutura. Universidade Federal de Uberlândia, 2019.

DOCKER-INC. Developers love docker.businesses trust it (2022). In: **Docker Inc (2022)**. [s.n.], 2022. Disponível em: <https://www.docker.com/>. Acesso em: 10.6.2022.

DOJOT, L. C. B. Dojot uma plataforma iot. In: **Dojot Linux CPQD Brasil (2022)**. [s.n.], 2022. Disponível em: <https://dojot.com.br/sobre-a-dojot-iot/>. Acesso em: 8.2.2022.

FULMARI, A.; CHANDAK, M. B. A survey on supervised learning for word sense disambiguation. In: **A survey on supervised learning for word sense disambiguation**. [s.n.], 2014. Disponível em: <https://pdfs.semanticscholar.org/58bb/8f4b9a0e7257ca15555e505e9fd35992f66c.pdf>. Acesso em: 10.6.2013.

FURQUIM, G. A. d. et al. Uma abordagem tolerante a falhas para a previsão de desastres naturais baseada em iot e aprendizado de máquina. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

GALLIZZI, B. Written by ben gallizzi, senior content editor - energy and electric vehicles. In: **Written by Ben Gallizzi, Senior Content Editor - Energy and Electric Vehicles (2022)**. [s.n.], 2022. Disponível em: <https://www.uswitch.com/gas-electricity/global-warming-and-natural-disasters/>. Acesso em: 4.6.2022.

JAISWAL, J. K.; SAMIKANNU, R. Application of random forest algorithm on feature subset selection and classification and regression. In: IEEE. **2017 world congress on computing and communication technologies (WCCCT)**. [S.l.], 2017. p. 65–68.

JUNIOR[1], A. A. de J.; MORENO[1], E. D. Segurança em infraestrutura para internet das coisas. 2015.

KUURKOVá, V. et al. **Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4 to 7, 2018, Proceedings, Part III**. [S.l.]: Springer, 2018. v. 11141.

MACHADO, G.; MENDOZA, M. R.; CORBELLINI, L. G. What variables are important in predicting bovine viral diarrhea virus? a random forest approach. **Veterinary research**, Springer, v. 46, n. 1, p. 1–15, 2015.

MARTINEZ, M. C. B. Maior perda em mariana e brumadinho foi de vidas humanas. In: **Maior perda em Mariana e Brumadinho foi de Vidas Humanas (2022) USP Universide de São Paulo**. [s.n.], 2022. Disponível em: <https://jornal.usp.br/atualidades/maior-perda-em-mariana-e-brumadinho-foi-de-vidas-humanas-diz-especialista/>. Acesso em: 4.6.2022.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole, v. 1, n. 1, p. 32, 2003.

MOSQUITTO. Eclipse mosquitto an open source mqtt broker (2022). In: **Eclipse Mosquitto An open source MQTT broker (2022)**. [s.n.], 2022. Disponível em: <https://mosquitto.org/>. Acesso em: 10.6.2022.

MUELLER, J. P.; MASSARON, L. **Aprendizado de Máquina para Leigos**. [S.l.]: Alta Books Editora, 2019.

OPENSOURCE, D. S. T. Tcpdump e libpcap (2022). In: **TCPDump e LibPCap (2022)**. [s.n.], 2022. Disponível em: <https://www.tcpdump.org/>. Acesso em: 10.6.2022.

PAN, S. J. Transfer learning. **Data Classification: Algorithms and Applications**, v. 21, 2014.

PAN, Y. et al. Pan sj, yang q. **A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010.

PASQUINI, R. et al. Admits: Architecting distributed monitoring and analytics in iot-based disaster scenarios. In: **Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva**. Porto Alegre, RS, Brasil: SBC, 2020. p. 11–20. ISSN 2595-6183. Disponível em: <https://sol.sbc.org.br/index.php/sbcup/article/view/11207>.

_____. Admits: Architecting distributed monitoring and analytics in iot-based disaster scenarios. In: SBC. **Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva**. [S.l.], 2020. p. 11–20.

PASQUINI, R.; STADLER, R. Learning end-to-end application qos from openflow switch statistics. In: **IEEE Conference on Network Softwarization (NetSoft)**. Porto Alegre, RS, Brasil: SBC, 2017. p. 11–20. ISSN 1-9.

PEREZ, R. R.; PEREIRA, F. C. Internet das coisas (iot). **PROJETOS E RELATÓRIOS DE ESTÁGIOS**, v. 1, n. 1, p. 1–40, 2019.

POLIKAR, R. Ensemble learning. In: **Ensemble machine learning**. [S.l.]: Springer, 2012. p. 1–34.

PYTHON, L. Linguagem python. In: **Linguagem Python (2022)**. [s.n.], 2022. Disponível em: <https://www.python.org/>. Acesso em: 10.6.2022.

QUINCOZES, S.; EMILIO, T.; KAZIENKO, J. Mqtt protocol: fundamentals, tools and future directions. **IEEE Latin America Transactions**, IEEE, v. 17, n. 09, p. 1439–1448, 2019.

REIS, P. A. d. Identification of vulnerable areas the floods and floods in urban areas through topographical and hydraulic models. In: **IEEE Conference on Network Softwarization (NetSoft)**. [S.l.]: SBC, 2015. p. 11–20. ISSN 127.

ROSA, A. Internet of things (iot). 2020.

SACCO, A. et al. An architecture for adaptive task planning in support of iot-based machine learning applications for disaster scenarios. **Computer Communications**, v. 160, p. 769–778, 2020. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366419316779>.

SÁNCHEZ, L.; COUSO, I.; CASILLAS, J. Genetic learning of fuzzy rules based on low quality data. **Fuzzy Sets and Systems**, Elsevier, v. 160, n. 17, p. 2524–2552, 2009.

SCIKIT, S. learn. Biblioteca sky-learn (2022). In: **Biblioteca Sky-learn (2022)**. [s.n.], 2022. Disponível em: <https://www.sky-learn.com/>. Acesso em: 10.6.2022.

SHAHABI, H. et al. A semi-automated object-based gully networks detection using different machine learning models: a case study of bowen catchment, queensland, australia. **Sensors**, MDPI, v. 19, n. 22, p. 4893, 2019.

SHEIBANIFARD, A.; SHAHROOD, I.; POUYAN, A. A. A conceptual architecture for disaster management using iot cloud. 2021.

SHENG, L. et al. Classification of iron ores by laser-induced breakdown spectroscopy (libs) combined with random forest (rf). **Journal of Analytical Atomic Spectrometry**, Royal Society of Chemistry, v. 30, n. 2, p. 453–458, 2015.

SILVA, F. D. L. d. Implementando adaptação em ambientes inteligentes utilizando sistemas multiagentes e aprendizado por reforço. 2019.

SOFTWARE, J. N. D. Jupyter notebook (2022). In: **Jupyter Notebook (2022)**. [s.n.], 2022. Disponível em: <https://jupyter.org/>. Acesso em: 10.6.2022.

STADLER, R.; PASQUINI, R.; FODOR, V. Learning from network device statistics. **Journal of Network and Systems Management**, Springer, v. 25, n. 4, p. 672–698, 2017.

TC, S. O. Tc traffic control. In: **TC Traffic Control (2022)**. [s.n.], 2022. Disponível em: <https://trafficcontrol.apache.org/>. Acesso em: 10.6.2022.

TOHRY, A. et al. Variable importance assessments of an innovative industrial-scale magnetic separator for processing of iron ore tailings. **Mineral Processing and Extractive Metallurgy**, Taylor & Francis, v. 131, n. 2, p. 122–129, 2022.

TRACEROUTE, S. O. Traceroute software opensource. In: **Tracerout Software Opensource (2022)**. [s.n.], 2022. Disponível em: <https://www.ibm.com/docs/pt-br/power8?topic=commands-traceroute-command>. Acesso em: 10.6.2022.

VANNUCHI, C. Noticias tragedia de brumadinho completa 3 anos (2022). In: **Noticias Atualizada (2022)**. [s.n.], 2022. Disponível em: <https://noticias.uol.com.br/cotidiano/ultimas-noticias/2022/01/25/>. Acesso em: 10.6.2022.