Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

LUCAS EMANUEL BATISTA DOS SANTOS

MIDDLEWARE BASEADO EM APRENDIZADO FEDERADO PARA APLICAÇÕES DE SMART CAMPUS

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA

2024

Lucas Emanuel Batista dos Santos

Middleware baseado em Aprendizado Federado para Aplicações de Smart Campus

Trabalho apresentado como requisito para a obtenção do título de Mestre, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal da Paraíba - IFPB.

Orientador: Prof. Dr. Ruan Delgado Gomes
Coorientador: Prof. Dr. Paulo Ribeiro Lins Júnior

João Pessoa
2024

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha - *Campus* João Pessoa, PB.

S237m Santos, Lucas Emanuel Batista dos.

Middleware baseado em aprendizado federado para aplicações de Smart Campus / Lucas Emanuel Batista dos Santos. - 2024.

147 f. : il.

Dissertação (Mestrado em Tecnologia da Informação) – Instituto Federal de Educação da Paraíba / Programa de Pós-Graduação em Tecnologia da Informação (PPGTI), 2024.

Orientação : Prof. D.r Ruan Delgado Gomes.

Coorientação : Prof. D.r Paulo Ribeiro Lins Júnior.

1. Middleware. 2. Aprendizado federado. 3. Privacidade. 4. Campus inteligente. 5. Internet das coisas. I. Título.

CDU 004.41(043)

Bibliotecária responsável Lucrecia Camilo de Lima – CRB 15/132

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

MESTRADO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO

LUCAS EMANUEL BATISTA DOS SANTOS

MIDDLEWARE BASEADO EM APRENDIZADO FEDERADO PARA APLICAÇÕES DE SMART CAMPUS

Dissertação apresentada como requisito para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós- Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB - Campus João Pessoa.

Aprovado em 01 de julho de 2024

Membros da Banca Examinadora:

Dr. Ruan Delgado Gomes

IFPB - PPGTI

Dr. Paulo Ribeiro Lins Júnior

IFPB - PPGTI

Dr. Paulo Ditarso Maciel Júnior

IFPB - PPGTI

Dra. Elloá Barreto Guedes da Costa

UEA

João Pessoa/2024

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, pela Sua Graça derramada e pelo privilégio de vivê-la e desfrutá-la, mesmo sem qualquer merecimento.

Além de agradecer, dedico também esta dissertação a minha família: Minha mãe, e pedagoga, Luzenira Batista, meu pai, e professor, Jedaías Pereira e a minha irmã, e mestre, Eliete Samara. Sem o auxílio, cuidado e o amor de vocês não seria possível ser quem sou hoje.

Meu orientador Prof. Dr. Ruan Delgado Gomes, meu coorientador Prof. Dr. Paulo Ribeiro Lins Júnior, professores e companheiros desde a graduação, agradeço principalmente toda a paciência e dedicação em todos esses anos. Ao IFPB, por ter sido minha segunda casa desde o ensino médio, além de grato, sinto-me orgulhoso de ter meu nome dentre os que puderam trilhar o caminho da educação nesta instituição.

Meu amigo e padrinho de casamento, Daniel Adonis, agradeço pelo incentivo e companhia desde a graduação. Por ter sido suporte nessa caminhada, até nas vezes que nem imagina.

A todas as pessoas que auxiliaram e incentivaram, desde colegas de trabalho, familiares e amigos, cada parcela, mesmo mínima, saibam que foi de grande importância.

Meus bichos de estimação, cachorra Pipoca e gato Chico, que tantos carinhos pediram (e receberam) durante essa trajetória. Ambos foram divertidíssimas companhias durante o desenvolvimento deste trabalho.

Por fim, reservo o agradecimento final e maior dedicação a minha esposa, Maria Isabelly. Seu amor, carinho, incentivo e risadas constantes com toda certeza tornaram esse caminho mais leve. Agradeço por você ser presente em todos os seus sentidos, de ser o presente, se fazer presente e ser o maior presente de Deus na minha vida. Te amo.

RESUMO

O crescente volume de dados em ambientes inteligentes apresenta uma oportunidade para utilizar o aprendizado de máquina para aprimorar a tomada de decisão e eficiência. No entanto, os métodos tradicionais de aprendizado de máquina enfrentam dificuldades com dados distribuídos, como também preocupações com a privacidade dos dados. O aprendizado federado (AF) oferece uma solução promissora, possibilitando aprendizado colaborativo distribuído sem comprometer a privacidade. Esta dissertação propõe o SFMEI (*Smart Federated Middleware for Educational Institutions*, um middleware projetado para intermediar o AF em aplicações de campus inteligente. O SFMEI apresenta vantagens para aplicações de campus inteligente, permitindo colaboração segura e com preservação de privacidade entre dispositivos e instituições, facilitando o desenvolvimento de modelos de aprendizado de máquina escaláveis, promovendo a integração de soluções de campus inteligente. O SFMEI adota uma arquitetura que permite a integração de vários algoritmos de aprendizado de máquina e modelos de dados distribuídos. Ele fornece uma API pública para simplificar a integração com soluções de campus inteligente e garantir colaboração segura e privacidade de dados entre os nós participantes. Os resultados experimentais demonstram a viabilidade do SFMEI em preservar o desempenho preditivo ao mesmo tempo em que aborda as questões de privacidade de dados. Para avaliar a eficácia do SFMEI, foi realizada uma avaliação experimental utilizando um modelo LSTM para previsão de séries temporais de dados de CO_2, Temperatura, Consumo de Energia e Consumo de Água. Os resultados demonstraram que o SFMEI não perde eficiência em comparação com uma abordagem de aprendizado de máquina não federada, atingindo um escore R^2 de 0.9898, contra 0.9895 da abordagem não federada. Também foi realizada uma avaliação para comparar algoritmos de agregação de AF, FedAVG e FedSGD, revelando que o FedSGD superou o FedAVG. Com o algoritmo FedAVG, cerca de $\pm 35\%$ dos modelos gerados foram classificados como "suficientes" ($R^2 > 0.75$), em comparação com $\pm 47\%$, utilizando algoritmo FedSGD. O SFMEI com o FedSGD atingiu um mediana do escore R^2 de 0.8639, em comparação com 0.8466 com o FedAVG. Um cenário simulado onde os clientes X, Y, Z consomem um modelo pré-treinado também apresenta resultados satisfatórios. Um modelo pré-treinado no SFMEI foi utilizado por três clientes hipotéticos, que não participaram do AF, resultando em um escore R^2 de 0.762735 para o cliente X, 0.644880 para o cliente Y e 0.763156 para o cliente Z.

Palavras-chaves: Middleware. Aprendizado Federado. Privacidade. Campus Inteligente. Internet das Coisas. IoT.

ABSTRACT

The growing volume of data in smart environments presents an opportunity to leverage machine learning for better decision-making and improved efficiency. However, traditional machine learning methods struggle with distributed data and data privacy concerns. Federated learning (FL) offers a promising solution, enabling collaborative learning without compromising privacy. This dissertation proposes the SFMEI (Smart Federated Middleware for Educational Institutions), a middleware specifically designed for FL in smart campus applications. The SFMEI presents advantages for smart campus applications, enabling secure and privacy-preserving collaboration among devices and institutions, facilitates the development of scalable machine learning models, and promotes the integration of smart campus solutions. SFMEI adopts an architecture that's allows integration of various machine learning algorithms and distributed data models. It provides a public API for easy integration with smart campus solutions and ensures secure collaboration and data privacy among participating nodes. Experimental results demonstrates the viability of SFMEI in preserving predictive performance while addressing data privacy concerns. To evaluate the effectiveness of SFMEI, an experimental evaluation was conducted using a LSTM model for time series forecast of CO_2, Temperature, Energy Consumption and Water Consumption datasets. The results demonstrated that SFMEI, do not lose effectiveness compared with a non-federated machine learning approach, achieved R^2 score of 0.9898, compared to 0.9895 of the non-federated. Also was conducted an evaluation for comparing federated learning aggregation algorithms, FedAVG e FedSGD, revealing that FedSGD outperformed FedAVG. With the FedAVG algorithm, $\pm 35\%$ of the generated models were classified as "sufficient" ($R^2 > 0.75$), while $\pm 47\%$ with the FedSGD algorithm. The SFMEI with the FedSGD achieved the median of R^2 score of 0.8639, compared to 0.8466 with FedAVG. A simulated scenario where clients X, Y, Z consumes a pre-trained model also presents satisfactory results. A pre-trained model on SFMEI was used by three hypothetical clients, that's not participated of FL, resulting in a R^2 score of 0.762735 for client X, 0.644880 for client Y and 0.763156 for client Z.

Key-words: Middleware. Federated Learning. Privacy. Smart Campus. Internet of Things. IoT.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things* (Internet das Coisas)
ZB	Zettabyte
PB	Petabyte
IA	Inteligência Artificial
FedIoT	*Federate Learning for Internet of Things* (Aprendizado Federado para Internet das Coisas)
CPU	*Central Process Unit* (Unidade Central de Processamento)
GPU	*Graphics Processing Units* (Unidades de Processamento Gráfico]
SFMEI	*Smart Federated Middleware for Educational Institutions* (*Middleware* de Federação Inteligente para Instituições de Ensino)
SLA	*Service Level Agreement* (Acordo de Nível de Serviço)

SUMÁRIO

1 INTRODUÇÃO

A internet das coisas (*Internet of Things* - IoT) relaciona-se diretamente com a transformação de ambientes em ambientes inteligentes, como, por exemplo, a transformação de cidades em cidades inteligentes,, as chamadas *smart cities* (Tan; Wang, 2010). As *smart cities* empregam tecnologias de IoT em soluções ligadas à mobilidade urbana, dispositivos móveis inteligentes, veículos e a gestão do tráfego (Brik; Ksentini; Bouaziz, 2020), mas não limita-se apenas a estas aplicações. São também desenvolvidas soluções voltadas a construções, no caso de prédios e casas inteligentes (Mitra; Ngoko; Trystram, 2021; Dasari et al., 2021) e também governança, o que ajuda a compreender e empregar um consumo inteligente de recursos nestas construções (Zanella et al., 2014). Baseados nesta pluralidade, há alguns anos observa-se o desenvolvimento de pesquisas norteadoras relacionadas à transformação e o desenvolvimento de *smart cities* (Zanella et al., 2014; Bennis; Debbah; Poor, 2018).

Neste contexto de *smart cities* podemos tomar, como exemplo, o estudo sobre técnicas de detecção de anomalias no consumo de energia, contribuindo para o aprimoramento da gestão deste recurso (Sedjelmaci; Senouci; Al-Bahri, 2016; Sater; Hamza, 2021; Taïk; Cherkaoui, 2020). Este cenário exemplifica como estudos como este, com foco em *smart buildings*, evoluíram a ponto to possibilitar a aplicação em larga escala, em contextos maiores, ou seja, em uma edificação de tamanho maior, contribuindo com a evolução do desenvolvimento de *smart cities* (Zanella et al., 2014; Mocanu et al., 2016).

Entretanto, a evolução e o desenvolvimento de tecnologias IoT carrega consigo preocupações (Tan; Wang, 2010) e também traz à tona fenômenos antigos, como o "dilúvio de dados", devido ao seu rápido crescimento e sua expansão no decorrer dos anos (Hey; Trefethen, 2003). Neste sentido, a segurança e privacidade dos dados gerados são preocupações que vem sendo exploradas no decorrer dos anos (Tan; Wang, 2010; Aïvodji; Gambs; Martin, 2019; Cao et al., 2020; Dasari et al., 2021), juntamente ao volume de dados gerado, algo já alertado há quase duas décadas (Hey; Trefethen, 2003).

O termo "dilúvio de dados", ou "explosão de dados", do inglês *data deluge*, foi utilizado para nomear o fenômeno do crescimento exponencial dos dados produzidos pela nossa sociedade atual ao passar dos anos, a partir dos seus mais diversos dispositivos (Hey; Trefethen, 2003). Em 2020 houve uma previsão de que em 2025 seriam gerados em torno de 73.1 Zettabytes (ZB) de dados (IDC, 2020). Entretanto, em 2022 a previsão foi atualizada, sendo previsto que a quantidade de dados gerados tende a dobrar a cada dois anos, sendo assim, uma nova expectativa é de que em 2025 a quantidade de dados gerados alcancem 176 ZB (IDC, 2022). Observando esse comportamento, o volume de dados gerados tende a crescer cada vez mais com o passar dos anos, de maneira diretamente ligada ao crescimento de soluções IoT, o que pode tornar estas

expectativas e previsões rapidamente defasadas.

Realizar análises a partir deste volume massivo de dados é algo imprescindível e o uso a Inteligência Artificial (IA) viabiliza essas análises e também a realização de predições, agregando valor a soluções IoT (Al-Garadi et al., 2020). Obter informações com base nestes dados tem sido há anos uma preocupação constante, para que esse volume massivo de dados não venha a se tornar inútil (Hey; Trefethen, 2003; Ahmed et al., 2017). Uma das formas de fazer isso é por meio do emprego de técnicas de IA, tipicamente com aprendizado de máquina, que necessitam de um volume massivo de dados para funcionar adequadamente. Esses dados devem ser referentes a uma diversidade de contextos e variáveis, que contemplem as diferentes situações que são importantes para a construção do modelo (Al-Garadi et al., 2020; Vemuri, 2019).

Não obstante, muitas vezes esses dados advém de fontes diferentes, com diferentes características de origem, de localização e até de disponibilidade. Essa situação é mais comum ainda quando se olha para o cenário típico de redes IoT com grande quantidade de nós, considerando sua distribuição geográfica, suas funcionalidades e até as características locais das variáveis monitoradas. Treinar um modelo de aprendizado de máquina a partir da fusão desses dados oriundos de múltiplas fontes pode ser uma tarefa difícil, a depender da escala da rede (Poornima; Paramasivan, 2020), sendo esta uma linha de pesquisa latente nos últimos anos, buscando lidar com a heterogeneidade de dados empregando técnicas de aprendizado de máquina (Li; Wang, 2019; Li et al., 2020; Kim; Wu, 2021; Wang et al., 2021; Shaer; Shami, 2023).

1.1 Motivação e Definição do Problema

De maneira direta, o emprego de IoT em ambientes correlatos a *smart cities* também pode fazer com que estes ambientes passem pelas mesmas preocupações e também implementem a mesma natureza de soluções. Conforme definido em (Chagnon-Lessard et al., 2021), observamos que *smart campus* é um ambiente inteligente que se relaciona de maneira direta com *smart cities*, posicionando-se entre o desenvolvimento de *smart buildings* e *smart cities*, como ilustrado na Figura 1. Assim, quando observamos que é possível estimar que sejam gerados, em um campus universitário, em torno de 1 petabyte (PB) de dados por ano (Chagnon-Lessard et al., 2021), vemos que os fenômenos relacionados a esses dados não são exclusivos de *smart cities*, ou grandes cenários de redes IoT, sendo também observados neste ambiente correlato de *smart campus* (Shi et al., 2021). Juntamente a isso, também destaca-se a importância do desenvolvimento de soluções voltadas para este ambiente inteligente (Chagnon-Lessard et al., 2021).

Somados estes aspectos ao fato de que, em parte, os dados de ambientes educacionais podem ser confidenciais (e.g. informações sobre os estudantes), uma metodologia específica para a construção de modelos de aprendizado de máquina, denominada de aprendizado federado, tem se demonstrado uma boa solução para o desenvolvimento de soluções inteligentes que utilizam

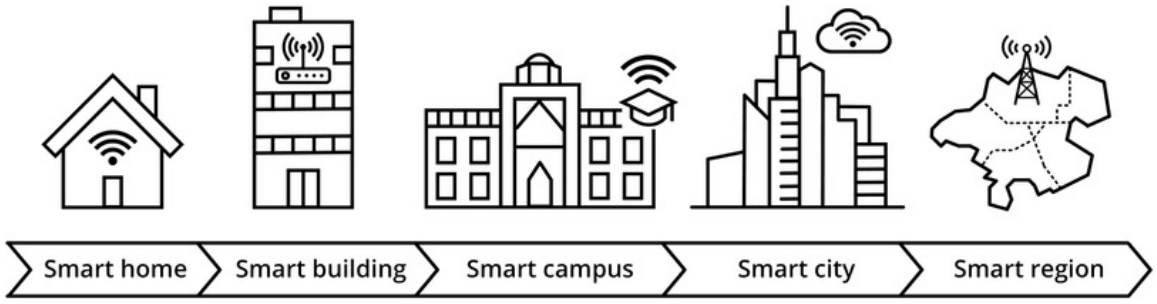


Figura 1 – Escala de *smartness* ("inteligência")

Fonte: Adaptado de (Chagnon-Lessard et al., 2021)

informações confidenciais oriundas de diferentes instituições/*campi*/departamentos (Mocanu et al., 2016). O aprendizado federado apresenta-se como uma forma de construção de modelos preditivos para sistemas de IoT, propondo o uso distribuído dos algoritmos de aprendizado de máquina (Bonawitz et al., 2019; Khan et al., 2021), com capacidade de garantir o anonimato dos dados dos nós finais (Behera et al., 2020), agregando valor às soluções de IoT atuais e futuras, de ambientes inteligentes, como *smart cities*, *smart buildings* e *smart campus* (Muhamad et al., 2017; Al-Garadi et al., 2020; Wu et al., 2020; Kaymakci; Baur; Sauer, 2021; Khan et al., 2021; Augusto, 2021; Guo, 2022; Monti et al., 2022; Brand et al., 2022; Alam et al., 2020; Lv, 2022).

Aprendizado federado é uma abordagem colaborativa e distribuída, sem a necessidade de compartilhamento dos dados usados no treinamento do modelo, proposta para a construção de sistemas inteligentes, com aprimoramento da privacidade (Konečnỳ et al., 2016; Kairouz et al., 2019; Nguyen et al., 2021). Segundo a definição de (Kairouz et al., 2019), "Aprendizado federado é uma aplicação de aprendizado de máquina em que vários clientes colaboram na solução de um problema de aprendizado de máquina, sob a coordenação de um servidor central do provedor de serviços".

Assim, este trabalho propõe um novo *middleware* chamado *Smart Federated Middleware for Educational Institutions (SFMEI)*. Ele foi idealizado para ser usado em ambientes educacionais inteligentes, permitindo a fácil integração de novas aplicações, independentemente da localização geográfica (Aydin; Sari; Oktug, 2023). Utilizando aprendizado federado para garantir a privacidade, o *SFMEI* visa possibilitar a análise, geração de *insights* e previsões de cenários, sem a necessidade de compartilhar dados locais. O *SFMEI* busca facilitar o desenvolvimento de soluções *smart campus*, permitindo adaptações sem exigir conhecimento específico dos algoritmos de aprendizagem federada. Assim, o objetivo principal é contribuir para a transformação de ambientes educacionais em ambientes inteligentes. Este estudo e o *SFMEI* buscam ser um recurso para viabilizar outros estudos e soluções práticas em *smart campus*, preenchendo uma lacuna identificada por (Freire, 2023).

O *SFMEI* parte de semelhanças com propostas já existentes (Guo; Zeng; Dong, 2020; Zhang et al., 2021b), entretanto, neste trabalho, se diferencia ao propor uma interface pública,

para que um cliente se conecte ao *middleware*, como meio de acesso a uma abstração com duas possibilidades de casos de usos:

- Case de uso 1: Colaborar de maneira participativa, utilizando seus dados locais para treinamento de modelos locais que posteriormente podem participar das rodadas de aprendizagem federada através do *SFMEI*;
- Case de uso 2: Fazer uso do *SFMEI* como repositório de modelos pré-treinados, que possibilitaria gerar predições de cenários locais, mesmo que não participando das rodadas de aprendizagem

Como insumos para validação de sua funcionalidade é proposto a utilização de dados públicos, relacionados a estudos já realizados para o desenvolvimento de soluções de *smart campus*. Como principais dados de validação, devido à recente publicação deste conjunto de dados e poucas pesquisas relacionadas, propões-se utilizar medidas de emissão de CO_2 (Eldeeb; Alves, 2023). Juntamente, também propõe-se o uso de dados de medidas de temperatura (Makonin et al., 2016), consumo de água (Weng; Zhang; Xia, 2018) e consumo de energia elétrica (Moraliyage et al., 2022), como parte de conjuntos de dados de apoio. Assim simulando o seguinte ecossistema ilustrado na Figura 2.

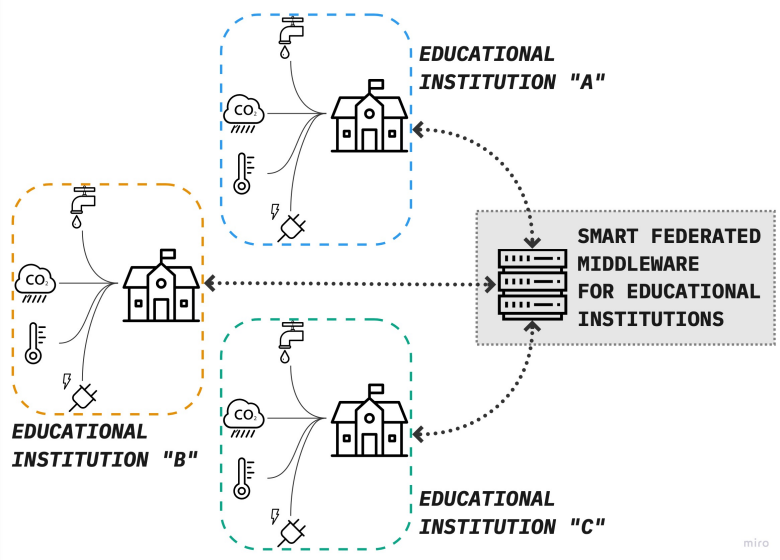


Figura 2 – Estrutura proposta do *middleware*

Fonte: Produzida pelos próprios autores

1.2 Delimitação do Trabalho

Este trabalho se concentrou especificamente em propor uma arquitetura e estrutura que suporte novas e futuras soluções de *smart campus* que utilizem aprendizado de máquina federado. O seu foco, quanto aos dados gerados e obtidos, é em compreender o comportamento

do *middleware* e seu impacto durante o treinamento de modelos de aprendizado de máquina de maneira federada. No que diz respeito aos resultados, este trabalho não visa obter resultados melhores que todos os demais estudos existentes atualmente em *smart campus*, mas espera-se que com os resultados seja possível demonstrar o *middleware* como uma solução relevante. Quanto aos algoritmos e modelos de aprendizado de máquina, este trabalho não visa gerar modelos de classificação, mas sim modelos de regressão utilizando *Long-Strong-Term Memory* (LSTM). Por esse motivo, métricas como *Confusion Matrix*, *Accuracy*, *Precision*, *Recall*, ou *F1 Score* não foram analisadas (Harrison, 2019). Este trabalho tem como o intuito de realizar predições em séries temporais, ou seja, prever valores numéricos contínuos futuros, métricas como essas são projetadas para avaliar a precisão de previsões, tipicamente em classes, para modelos de classificação. Por fim, esse trabalho não visa funcionar como uma solução de *smart campus* para uma determinada instituição de ensino específica, mesmo que possa ser adotada, ou replicada, por alguma instituição em algum momento futuro.

1.3 Contribuições

As contribuições deste trabalho podem ser divididas em um conjunto de contribuições principais, para soluções em *smart campus*, e em contribuições secundárias, para o desenvolvimento de um *middleware* de aprendizado federado. De maneira detalhada estas contribuições são citadas nos seguintes tópicos desta seção.

1.3.1 Para soluções em *Smart Campus*

Este trabalho visa seguir contribuindo e corroborando com os recentes estudos para a aceleração e viabilização de soluções em *smart campus* (Aydin; Sari; Oktug, 2023; Eldeeb; Alves, 2023; Corradini et al., 2023), abrangendo diversas categorias de estudo. Detalhadamente, quanto às contribuições principais deste trabalho, podemos classificá-las da seguinte maneira:

- A $\rightarrow$ Quanto à proposição de um *middleware*;
- B $\rightarrow$ Quanto aos conjuntos de dados utilizados;
- C $\rightarrow$ Quanto ao uso de aprendizado federado.

Na Figura 3 as contribuição são relacionadas conforme as categorias de estudos em *smart campus*, definidas por (Chagnon-Lessard et al., 2021). Por fim, este trabalho difere-se dos demais por propor uma solução de *smart campus* de maneira unificada e que possa abranger diversos possíveis estudos, de maneira que não realize apenas o treinamento de modelos, mas também que funcione como um repositório de modelos pré-treinados, relacionados ao contexto de *smart campus*.

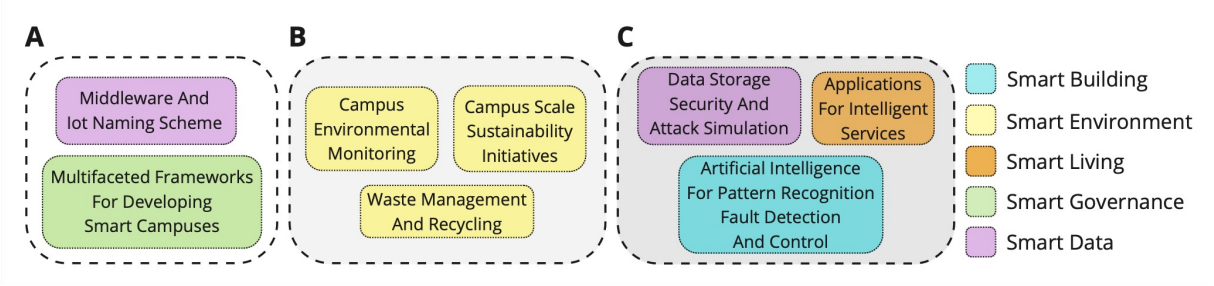


Figura 3 – Contribuições por categorias de pesquisa para o desenvolvimento de *smart campus*

Fonte: Produzida pelos próprios autores

Também pode-se, através de (Augusto, 2021), compreender um *smart campus* como um ambiente inteligente e sua relação com três grupos de *stakeholders* principais:

- Ensino e aprendizagem → Relativo a estudantes, professores;
- Pesquisa e Inovação → Relativo, por exemplo, a estudantes de pós-graduação e pesquisadores, mas também laboratórios e seus auxiliares, ou responsáveis;
- Tomada de decisão e Suporte → Relativo a diversos grupos de gestão, suporte operacional e organizacional, de todos os níveis.

Assim, a proposta deste trabalho agrega valor e pode contribuir de maneira direta para interesses de dois destes grupos de *stakeholders*: (i) Pesquisa e Inovação: quando toma o *SFMEI* como um facilitador para que soluções de *smart campus* sejam exploradas em temas de pesquisa e inovação; (ii) Tomada de decisão e Suporte: quando o *SFMEI* se torna solução que pode contribuir de maneira direta para, por exemplo, o gerenciamento de recursos ou suporte ao ensino, quando, respectivamente, disponibiliza um modelo de aprendizado de máquina para predição de consumo de energia, ou de risco de evasão de estudantes.

1.3.2 Para o desenvolvimento de um *middleware* de aprendizado federado

De maneira secundária, observando os estudos e propostas de *middlewares* e *frameworks* de aprendizado federado, conforme Tabela 1, foi observado que, embora existam estudos de propostas abrangentes, não foram identificados estudos e propostas de uma solução específica de *middleware* para *smart campus*. Através deste estudo é possível atingir essa área, de forma específica, com uma nova contribuição dentre estas soluções já existentes. E assim, conforme concluído por (Ahmed; Ramadhan; Elatab, 2022), pode-se desenvolver uma solução que aprimora o processo de análise de dados de *IoT*, fazendo uso de Inteligência Artificial e aprendizado de máquina, contribuindo para o desenvolvimento de *middlewares* de *IoT*.

Tabela 1 – *Middlewares* e *frameworks* de aprendizado federado e suas fontes

Middleware/Framework	Application	Public	Repository Address
CorrFL (Shaer; Shami, 2023)	Heterogeneity	YES	github.com/Western-OC2-Lab/CorrFL
Framework for 5G Networks (Liu et al., 2020)	Network	NO	
FedLab (Zeng et al., 2023)	Heterogeneity	YES	github.com/SMILELab-FL/FedLab
FLOWER (Beutel et al., 2022)	Large-scale Federated Learning Experiments	YES	github.com/adap/flower
FedPD (Zhang et al., 2021c)	Non-IID	YES	github.com/564612540/FedPD
PYVERTICAL (Romanini et al., 2021)	Federated Learning Approach Improvement	YES	github.com/OpenMined/PyVertical
FedCV (He et al., 2021)	Computer Vision	YES	github.com/FedML-AI/FedCV/
Agnostic Federated Learning (Mohri; Sivek; Suresh, 2019)	Fairness	NO	
FedLoc (Yin et al., 2020)	Data-Driven	NO	
FedMed (Wu; Liang; Wang, 2020)	Language Modeling	NO	
Federated Learning for Healthcare (Xu et al., 2021)	Healthcare	NO	
FedSpace (So et al., 2022)	Federated Continual Learning	YES	github.com/LTTM/FedSpace
Hetero-FedIoT (Khan et al., 2023)	Heterogeneity	NO	
OrderlessFL (Nasirifard; Mayer; Jacobsen, 2022)	Permission	YES	github.com/orderlesschain/orderlessfl
ModularFed (Arafeh et al., 2023)	Custom Learning	YES	github.com/arafeh94/ModularFed
FRAMH (Mazzocca et al., 2022)	Healthcare	NO	
Fleet (Damaskinos et al., 2020)	Online Learning	YES	github.com/gdamaskinos/fleet
Bristle (Verbraeken; Vos; Pouwelse, 2021)	Non-IID	NO	
FedIot (Zhang et al., 2021b)	General Purpose	YES	github.com/FedML-AI/FedML/tree/master/iot

1.4 Objetivos

O objetivo geral deste trabalho consistiu em projetar e implementar um *middleware* de aprendizado federado para *smart campus* com acoplamento simplificado de soluções.

Para alcançar plenamente este objetivo geral os objetivos específicos são:

1. Projetar o *middleware* proposto, sua arquitetura e sua *API* pública de maneira detalhada;
2. Implementar *middleware*, baseado na arquitetura proposta, para treinamento de modelos de aprendizado de máquina;
3. Levantar conjuntos de dados que sirvam como insumos de pesquisas realizadas na área de *smart campus* para que funcionem como dados de testes para *middlewares*;
4. Processar o conjunto de dados de maneira que possa ser utilizado como dados de treino para modelos de regressão no cenário de aprendizado federado;

5. Realizar uma avaliação experimental do *middleware* implementado, para validar a suas funcionalidades e verificação das métricas coletadas;

1.5 Questões de Pesquisa

Após essa breve contextualização acerca da motivação, problema, e apresentações dos objetivos, as seguintes questões de pesquisa foram levantadas para guiar o desenvolvimento deste trabalho:

- QP1) Há prejuízos em utilizar o *middleware* proposto em comparação com uma abordagem de aprendizagem de máquina não federada?
- QP2) É possível ao *middleware* proposto ter bom desempenho para os diferentes conjuntos de dados considerados neste trabalho?
- QP3) É possível ao *middleware* proposto ter bom desempenho em algoritmos distintos de aprendizagem federada?
- QP4) Algum dos algoritmos de agregação possui melhor desempenho que o outro?
- QP5) Um número reduzido de nós participantes no treinamento pode reduzir a capacidade de um modelo obter um escore alto?
- QP6) O *middleware* proposto é capaz de fornecer um modelo pré-treinado a ser utilizado para predizer cenários locais sem reduzir drasticamente sua performance?

1.6 Estrutura do Documento

Os capítulos subsequentes estão organizados da seguinte maneira:

No Capítulo 2, os conceitos de *Smart Campus*, aprendizado federado e *Middlewares* são apresentados em detalhes, juntamente com uma rápida explanação sobre e trabalhos relacionados, apresentando os estudos que representam o estado da arte relacionado aos objetivos e contribuições desta dissertação.

No Capítulo 3 é apresentada a metodologia utilizada no desenvolvimento da pesquisa, juntamente com uma breve descrição do processo de construção e desenvolvimento do protótipo do *middleware*, o procedimento de avaliação e as métricas coletadas nesse contexto.

No Capítulo 4 o *middleware* proposto é apresentado, incluindo sua arquitetura, casos de uso e funcionamentos básicos.

No Capítulo 5 estão presentes os resultados obtidos a partir dos experimentos de validação, bem como uma discussão sobre estes resultados.

As considerações finais, as propostas de continuação do trabalho e as limitações da pesquisa são descritos no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os principais fundamentos teóricos que serviram como base para o desenvolvimento deste trabalho. Sendo eles um detalhamento acerca de *smart campus*, aprendizado federado e *middlewares* IoT, juntamente a uma apresentação dos estudos que representam o estado da arte relacionado ao tema deste trabalho, de acordo com o levantamento realizado para esta dissertação.

2.1 *Smart Campus*

O *smart campus* é um ambiente inteligente que pode gerar cerca de 1 petabyte de dados por ano (Chagnon-Lessard et al., 2021), indicando que a massiva quantidade e heterogeneidade dos dados também estão presentes nesse contexto, juntamente com seus desafios (Shi et al., 2021). Soluções IoT para *smart campus* oferecem diversos benefícios às instituições de ensino, como o aprimoramento da eficiência de custos, a criação de salas de aula inteligentes, o gerenciamento de recursos e a sustentabilidade (Cavus et al., 2022). No entanto, conforme apontado em (Giuriatti et al., 2023), ainda há falta de estudos práticos no desenvolvimento de *smart campus*, o que permite a exploração de novos contextos e a diversificação das soluções, trazendo ainda mais benefícios.

2.2 Aprendizado Federado

Aprendizado federado é uma abordagem de aprendizagem de máquina colaborativa e distribuída, proposta para a construção de sistemas inteligentes, com aprimoramento da privacidade, de maneira que não compartilhe os dados locais usados no treinamento dos modelos, além de reduzir a carga de comunicação, em relação ao volume de dados trafegados para treinamento dos modelos (Konečnỳ et al., 2016; Kairouz et al., 2019; Nguyen et al., 2021). Nessa abordagem, vários clientes colaboram na construção de um modelo de aprendizado de máquina, sob a coordenação de um servidor central. Os clientes colaboradores são chamados de *workers* e o servidor central de *aggregator* (Kairouz et al., 2019). A partir dessa definição, de maneira mais detalhada, e ilustrada na Figura 4, o funcionamento básico de um sistema de aprendizado federado se dá por meio dos seguintes passos:

1. o *aggregator* inicia um modelo global com parâmetros arbitrários;
2. os *workers* recebem o modelo global e calculam localmente um novo modelo com seus dados locais;
3. os *workers* enviam a atualização de seu modelo local para o *aggregator*;

4. o *aggregator* combina todos os modelos locais em um novo modelo global aprimorado;

5. o novo modelo global é avaliado e os *workers* recebem a atualização deste modelo e, assim, o processo se repete até que o treinamento global seja concluído.

Para finalização do treinamento é levado em consideração algum fator avaliativo definido no *aggregator*, podendo ser o valor da métrica ou *score* de avaliação do modelo, como também o número de rodadas de federação, ou seja, quantas vezes um modelo passará por todo o processo de aprendizagem, sendo reenviado aos clientes, em seguida agregado e avaliado. Pode-se observar que a informação trafegada durante o processo de aprendizado é relativa apenas aos dados dos modelos treinados em cada *worker* e no *aggregator*, minimizando riscos de vazamento de dados, mantendo ainda privados todos os dados utilizados durante o treinamento.

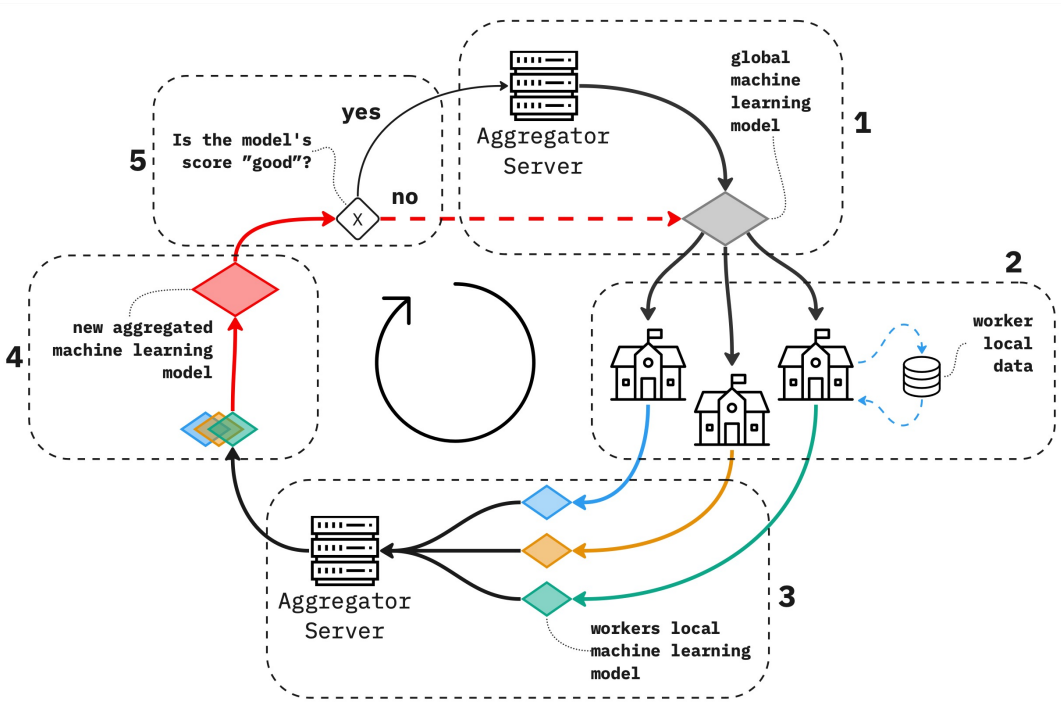


Figura 4 – Funcionamento básico de um sistema de aprendizado federado

Fonte: Produzido pelos próprios autores

2.2.1 Algoritmos de Aprendizagem Federada

No aprendizado federado, a agregação dos modelos treinados pelos dispositivos é uma das partes principais do processo de aprendizagem. Para realizar essa agregação de modelos de aprendizado federado alguns algoritmos são utilizados. Destacam-se dois como os algoritmos de agregação mais utilizados, o FedSGD e o FedAVG.

O FedSGD deriva-se do algoritmo (*stochastic gradient descent*), que tipicamente escolhe uma amostra parcial randômica dentro de um conjunto para analisar e dar seguimento ao treinamento de modelos de aprendizado de máquina. No aprendizado federado tipicamente é

induzido a "escolher"toda a mostra de uma rodada de treinamento, e buscar dentre esta o melhor modelo entre os enviados pelos dispositivos, priorizando a precisão. (McMahan et al., 2017) O *aggregator* classifica o melhor dentre os modelos treinados pelos *workers*, repassando esse modelo aos *workers* para que assim siga o treinamento nas próximas rodadas.

O FedAVG se destaca pela sua simplicidade. Sua proposta de funcionamento funciona de modo que, uma vez que o *aggregator* receba todos os modelos treinados pelos *workers*, ele aplica uma função simples de média entre os modelos calculados a cada rodada de federação (McMahan et al., 2017). Essa simplicidade do FedAVG resulta em eficiência, exigindo menos comunicação e computação do que outros algoritmos, apresentando também uma rápida convergência (Kontar et al., 2021).

Por conta da busca por maior precisão, o FedSGD exige mais comunicação e computação, além de apresentar convergência mais lenta do que o FedAVG, conforme estudo descrito em (Kontar et al., 2021). Entretanto, esta comunicação e computação maior pode levar a resultados mais precisos do que o FedAVG (McMahan et al., 2017).

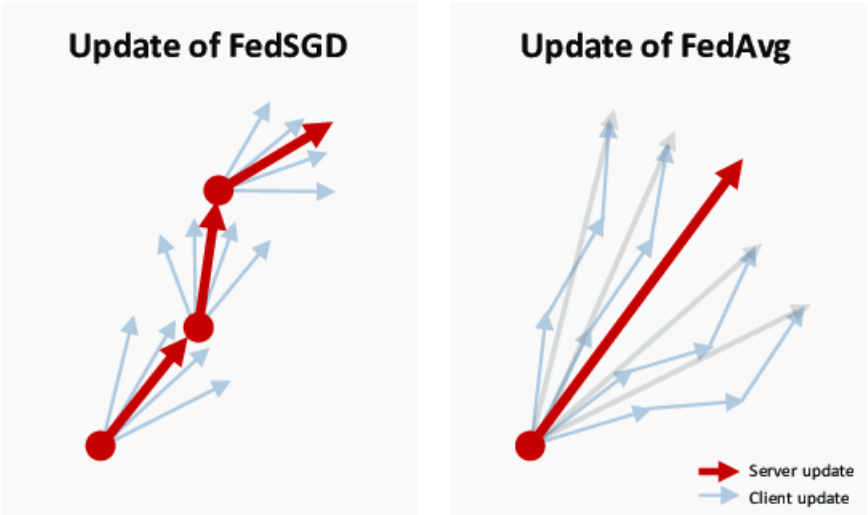


Figura 5 – Ilustração comparativa entre FedAVG e FedSGD.

Fonte: Adaptado de (Kontar et al., 2021)

A Figura 5, adaptada de (Kontar et al., 2021), ilustra o comparativo entre ambos os algoritmos. Do lado esquerdo da figura pode-se observamos a maneira em que o modelo melhor classificado é repassado a cada rodada de agregação do FedSGD. Da mesma maneira, no lado direito é possível observar o modelo resultante do cálculo da média realizado pela agregação através do FedAVG.

Mesmo sendo os mais comuns, estes algoritmos não são os únicos propostos para agregação em aprendizado federado. Nos últimos anos diversas outras propostas de novos algoritmos vêm sendo estudadas e exploradas. Em alguns casos, estudos que implementam variações e customizações de FedAVG e FedSGD demonstraram-se como soluções mais adaptáveis aos

cenários específicos abordados nos trabalhos (Nguyen et al., 2020; Hamer; Mohri; Suresh, 2020; Wu; Wang, 2021; Zeng et al., 2023; Zhang et al., 2023; Wu et al., 2023; Qi et al., 2023; Beltrán et al., 2023).

2.2.2 Estratégias de treinamento

O treinamento de um modelo de aprendizado federado é aplicado considerando uma abordagem de computação distribuída chamada computação de borda (*Edge Computing*). Esta abordagem aproxima o processamento e armazenamento de dados da localização onde eles são gerados, deste modo, em vez de de enviar todos os dados para servidores centralizados na nuvem o processo ocorre mais próximo a "borda"da rede, ou seja, perto dos dispositivos e sensores que produzem esses dados. De acordo com a capacidade computacional dessa "borda"é que distinguem-se duas estratégias distintas de treinamento de um modelo de um modelo de aprendizado federado:

1. ***Cross-Device***. O treinamento do modelo local é realizado diretamente no dispositivo que coleta os dados, como ilustrado na Figura 6, um *smartphone*. Nesta estratégia de treinamento o dispositivo final precisa possuir uma capacidade computacional suficiente para realizar o treinamento de modelos de aprendizado de máquina (Imteaj et al., 2021; Rehman et al., 2021; Gao et al., 2023);

2. ***Cross-Silo***. O treinamento do modelo local é realizado em um “Silo”, próximo aos dispositivos da borda da rede que coletam os dados, como ilustrado na Figura 7, onde os dados são gerados por sensores e armazenados em bases de dados. Nesta estratégia o dispositivo final pode ser capaz de apenas coletar os dados, sem necessidade de possuir grande poder computacional de processamento. A responsabilidade do treinamento é repassada para algum serviço presente no “silo” (Huang; Huang; Liu, 2022; Terrail et al., 2022).

Através da estratégia *Cross-Silo* é possível que domínio maiores, como, inclusive, um *smart campus*, sejam participantes do treinamento de modelos de maneira federada mantendo suas soluções internas privadas.

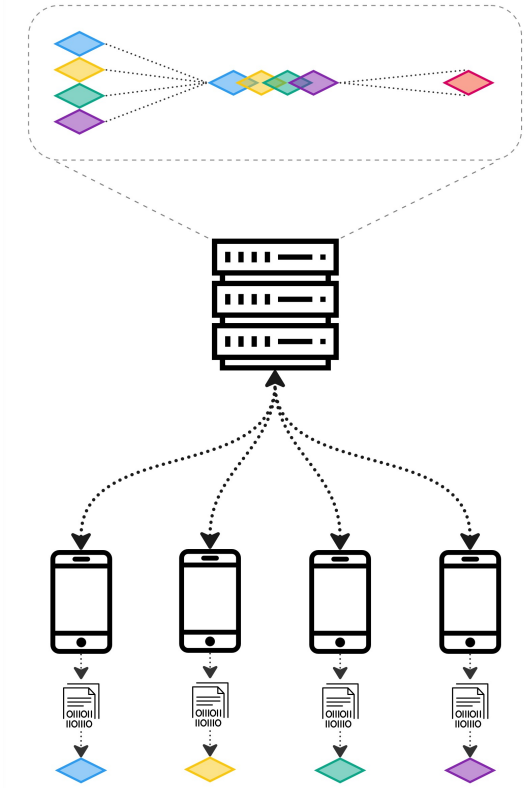


Figura 6 – Aprendizado federado *cross-device*

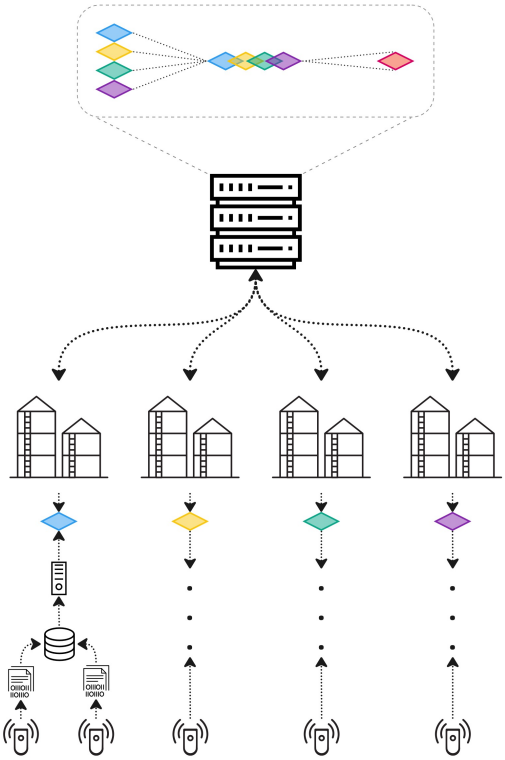


Figura 7 – Aprendizado federado *cross-silo*

Fonte: Produzido pelos autores

2.2.3 Aplicações

Uma solução de aprendizado federado, aplicada em uma rede de dispositivos IoT, utilizando o poder computacional destes diversos dispositivos, possibilita a transformação de ambientes em ambientes inteligentes, fornecendo privacidade aos dados coletados (Nguyen et al., 2021). Algumas das soluções para transformação de ambientes inteligentes possuem aplicações em alguns outros contextos. Como exemplo, pode-se listar:

- ***Smart Industry* (Indústria Inteligente)**. O aprendizado federado pode fornecer soluções em domínios como a robótica e a Indústria 4.0, mantendo a privacidade dos dados coletados no ambiente industrial (Zhou et al., 2018);
- ***Smart Healthcare* (Saúde Inteligente)**. Possibilita que sejam providas soluções eficientes para saúde inteligente (Cao et al., 2020), habilitando mudanças nos modelos nos atuais sistemas de saúde, melhorando a privacidade do usuário através da aplicação em distintas entidades do sistema de saúde, desde os usuários até os provedores (e.g. planos de saúde, hospitais e clínicas) (Rieke et al., 2020);
- ***Smart Transportation* (Transporte Inteligente)**. O aprendizado federado tem sido apli-

cado na borda da rede, através da sua aplicação nos veículos como nós finais, com finalidade de aprimorar e habilitar recursos que, de maneira colaborativa, treinam modelos de IA sem comprometer a privacidade dos usuários, ou veículos. Pode-se desenvolver soluções para contribuir, por exemplo, com o planejamento de tráfego, substituindo abordagens tradicionais de aprendizado de máquina centralizado em tarefas de previsão de tráfego, ao executar modelos diretamente no veículos, com base em seus conjuntos de dados (Lim et al., 2021).

2.3 *Middleware*

De maneira abrangente, um *middleware*, conforme definição de (Li et al., 2015), tem como intuito prover uma interface homogênea que abranja soluções genéricas para a gestão de algum contexto. Já em (Bandyopadhyay et al., 2011), é definido que o objetivo comum de um *middleware* é desenvolver um *framework* capaz de habilitar uma camada de aplicação de um modo *plug-n-play*. De acordo com (Razzaque et al., 2015), um *middleware* pode oferecer serviços comuns para aplicações e facilitar o desenvolvimento de aplicações, integrando dispositivos heterogêneos de computação e comunicação. De acordo com estas definições, pode-se assumir, para o contexto deste trabalho, que um *middleware* encaixa-se como uma camada capaz de que prover uma interface que facilite o desenvolvimento de aplicações de capacidades computacionais heterogêneas através de um acesso *plug-n-play*.

Em sistemas de *IoT*, um *middleware* pode ser classificados em três tipos, de acordo com seus contextos de gestão (Ahmed; Ramadhan; Elatab, 2022):

1. *Middleware* para gestão de dispositivos: gestão do ciclo de vida de dispositivos IoT, como conexão, configuração e segurança;
2. *Middleware* para gestão de dados: gestão dos dados gerados por dispositivos IoT, incluindo sua coleta, armazenamento e processamento;
3. *Middleware* para gestão de aplicações: Responsável por prover uma interface de acesso aos dados gerados por dispositivos IoT.

De acordo com a complexidade dos domínios envolvidos no contexto da solução, o *middleware* pode se relacionar de maneira transversal entre estes tipos, podendo ser ditos como *middlewares* híbridos (Razzaque et al., 2015).

2.4 Trabalhos Relacionados

As pesquisas voltadas para soluções em *smart campus* são comumente classificadas em diversas categorias. Conforme revisão descrita em (Chagnon-Lessard et al., 2021), sete categorias não excludentes foram identificadas, em que cada uma dessas categorias se relaciona com linhas

de pesquisa específicas que estão descritas na Tabela 2. Além disso, algumas dessas categorias podem conceitualmente correlacionar-se com outras, por meio de de pesquisas que possam abranger diversas linhas (Chagnon-Lessard et al., 2021; Guo, 2022; Monti et al., 2022; Brand et al., 2022; Alam et al., 2020; Lv, 2022).

Tabela 2 – Categorias de pesquisa para o desenvolvimento de *smart campus*

CATEGORIAS DE PESQUISA PARA ESTUDOS EM *SMART CAMPUS*	
Prédio Inteligente	Análise da capacidade de prédios e reforma de prédios
	Plataformas e *frameworks* de Big Data
	Integração de coleta de dados em tempo real no ambiente BIM (*Building Information Modeling*)
	Inteligência Artificial para reconhecimento de padrões, detecção de falhas e controle
	Papel/Responsabilidade dos ocupantes
Ambiente Inteligente	Microgrids
	Iniciativas de sustentabilidade para todo o campus
	Monitoramento ambiental do campus
Mobilidade Inteligente	Monitoramento e compreensão da mobilidade em campi
	Transporte de ônibus
	Assistência e navegação no campus
	Carregamento de veículos elétricos
	Estacionamento de carros
Convívio Inteligente	Aplicações para serviços inteligentes
	Mineração de dados para recomendação de experiências e serviços personalizados
	Sistemas de vigilância, localização e navegação
	Uso do espaço em tempo real
Pessoas Inteligentes	Compreendendo as percepções sobre inteligência
	Monitoramento da opinião das pessoas
	Transferência de conhecimento e colaboração
	Aprendizagem ubíqua
	Gamificação e ambientes de aprendizagem virtual
	Inovação estudantil
	Salas de aula e salas de conferência inteligentes
Governança Inteligente	*Frameworks* multifacetados para o desenvolvimento de campus inteligentes
	Campus virtuais para planejamento e simulação
	Sistemas de tomada de decisão e gerenciamento
	Certificações e *benchmarking*
	Práticas de dados abertos
Dados Inteligentes	Redes de sensores sem fio e infraestruturas sem fio
	Middleware e esquema de nomes para IoT
	Conscientização do contexto
	Segurança de armazenamento de dados e simulação de ataques

	Autenticação

Já no trabalho descrito em (Nóbrega; Chim-Miki; Castillo-Palacio, 2022), encontra-se uma categorização em oito dimensões, em que uma destas oito dimensões, a de *smart techology*, se relaciona de maneira transversal entre as demais sete.Na revisão de literatura descrita em (Giuriatti et al., 2023) também encontra-se uma categorização em seis dimensões *smart*, sendo elas: (i) Ambientes; (ii) Aplicações; (iii) Infraestrutura; (iv) Sustentabilidade; (v) Técnicas e (vi) Tecnologias. Embora bastante abrangentes, estas categorizações demonstram possíveis focos distintos. Nas revisões guiadas tanto por (Giuriatti et al., 2023), quanto também por (Nóbrega; Chim-Miki; Castillo-Palacio, 2022), pode-se observar um foco na categorização para o desenvolvimento estratégico, ou de gestão, de soluções de *smart campus*.

O trabalho descrito em (Chagnon-Lessard et al., 2021) aborda os aspectos tecnológicos de maneira mais específica, descrevendo tecnologias atuantes e importantes em cada categoria. Na Tabela 2 é possível observar linhas de pesquisas como "**Plataformas e *frameworks* de *Big Data***", "***Frameworks* multifacetados para o desenvolvimento de campus inteligentes**", "**Redes de sensores sem fio e infraestruturas sem fio**" e "***Middleware* e esquema de nomes para IoT**", o que demonstra um detalhamento técnico maior no trabalho descrito em (Chagnon-Lessard et al., 2021) em comparação aos estudos em (Nóbrega; Chim-Miki; Castillo-Palacio, 2022) e (Giuriatti et al., 2023). Assim, esta categorização pode ser assimilada como uma referência mais específica para a categorização de soluções e estudos mais técnicos em *smart campus*.

2.4.1 Aprendizado Federado e *Smart Campus*

Especificamente, é possível encontrar estudos e soluções para *smart campus* que aplicam aprendizado federado em distintos cenários escolares, como esporte (Sun, 2022), desempenho escolar (Guo; Zeng; Dong, 2020), ensino (Chen, 2023) e também gerenciamento de recursos (Zhang et al., 2021a).

Estes estudos propõem soluções voltadas a:

- Uma nova técnica (abordagem) de aprendizado federado aplicada a um conjunto de dados de práticas esportivas e sua qualidade (Sun, 2022);
- Um *framework*, FEEDAN, para a análise de desempenho escolar, analisando dois conjuntos de dados reais, por meio de duas técnicas distintas de aprendizado federado (Guo; Zeng; Dong, 2020). Entretanto, não especifica outros casos de uso, a não ser com o fim de treinamento de modelos;
- Um estudo e discussão acerca de como utilizar técnicas de aprendizado federado para detecção de sentimentos durante aulas online, afim de aprimorar o processo de ensino (Chen, 2023);

- Um estudo acerca de um modelo de aprendizado federado dentro de uma mesma instituição para aprimorar a predição do consumo de energia (Zhang et al., 2021a).

Apesar de apresentarem soluções para aplicações diversas, os trabalhos citados não descrevem nem propõem uma solução de *middleware* a ser aplicada no contexto de *smart campus*.

2.4.2 *Middlewares e Frameworks*

As propostas recentes de *middlewares*, como soluções que implementem aprendizado federado, possuem o intuito principal de melhorar o desempenho e a distribuição da aplicação do aprendizado federado em contextos específicos. Detalhadamente, foram identificadas soluções com foco em: (i) detecção de anomalias aplicada a *middleware* de aprendizado federado para o sistema operacional Android (Damaskinos et al., 2020); (ii) desenvolvimento de *middleware* de aprendizado federado descentralizado para ambientes de dados heterogêneos (Verbraeken; Vos; Pouwelse, 2021). Essas soluções não se relacionam de maneira direta como uma solução para *smart campus*.

Esta lacuna é observada também quando relacionamos outros estudos. Na Tabela 3, que pode ser encontrada completa no Apêndice A, são listadas diversas propostas de *middlewares* que implementem aprendizado federado, sejam ligados a soluções de *smart campus* ou não, além dos *middlewares* citados anteriormente. Conforme as categorias levantadas por (Chagnon-Lessard et al., 2021), foi realizada a categorização do *SFMEI* e dos estudos da Tabela 3, evidenciando que não foi possível identificar uma proposta específica de um *middleware* que se correlacione com outras categorias no desenvolvimento de soluções para *smart campus*. Com a finalidade de melhorar a legibilidade do documento, detalhamos a relação do SFMEI com outras propostas no Apêndice B, nas Tabelas 28 e 29. A relação destes trabalhos junto às contribuições do *SFMEI* detalhadas na seção , demonstram que o *SFMEI* distingue-se dos demais trabalhos encontrados principalmente pela maneira que explora múltiplas e distintas categorias de estudo em *smart campus*.

2.4.3 *Uma análise comparativa entre SFMEI e FedIoT*

O *SFMEI* basea-se a partir da análise realizada da implementação do *FedIoT*, proposto por (Zhang et al., 2021b) e disponível em um repositório público da *FedML Inc*.[1]. Entretanto, para a implementação do *FedIoT* se faz necessário um nível de conhecimento específico acerca das soluções técnicas que são utilizadas. O *SFMEI*, com sua proposta, inclui uma camada de abstração no cliente, nó final, para que, independente dos seus recursos locais, o cliente possa fazer uso das soluções disponíveis no *middleware*.

O *FedIoT* provê uma arquitetura que possibilita ser implementada de maneira generalista em sistemas e dispositivos IoT, entretanto é descrita e detalhada utilizando componentes e

[1] https://github.com/FedML-AI/FedML/tree/master/iot.

Tabela 3 – Tabela de referencia de estudos analisados em *Smart Campus* e aprendizado federado

Estudo	**Chave Referencial**	**Tipo**
1	*PedagogicalFL4.0*	*Smart Campus*
2	*SmartCampusFramework*	*Smart Campus*
3	*LoRaWan-SmartCampus*	*Smart Campus*
4	*FloWare*	Aprendizado Federado
5	*CorrFL*	Aprendizado Federado
6	*SecureFL5.0*	Aprendizado Federado
7	*FedLab*	Aprendizado Federado
8	*Flower*	Aprendizado Federado
9	*FedPD*	Aprendizado Federado
10	*Pyvertical*	Aprendizado Federado
11	*Fedcv*	Aprendizado Federado
12	*AgnosticFL*	Aprendizado Federado
13	*FedLoc*	Aprendizado Federado
14	*Fedmed*	Aprendizado Federado
15	*FLHealthcare*	Aprendizado Federado
16	*Fedspace*	Aprendizado Federado
17	*Hetero-FedIot*	Aprendizado Federado
18	*OrderlessFL*	Aprendizado Federado
19	*ModularFed*	Aprendizado Federado
20	*FRAMH*	Aprendizado Federado
21	*Fleet*	Aprendizado Federado
22	*Bristle*	Aprendizado Federado
23	*FLIoT*	Aprendizado Federado

dispositivos específicos. Devido à natureza da proposta do *SFMEI*, ele foi idealizado de modo que, em vez de considerar dispositivos IoT específicos como os nós finais, as instituições de ensino sejam consideradas os nós finais. O *SFMEI* propõe viabilizar que cada instituição de ensino possa ser participante da solução global, mesmo que, localmente, possuam soluções distintas e únicas para o desenvolvimento de suas soluções inteligentes.

Por exemplo, uma instituição de ensino IE_A pode estar desenvolvendo uma solução de *Smart Campus* que utilize a plataforma Arduino como dispositivo final para monitoramento e melhoria do consumo de energia. Esta instituição, IE_A, pode ser participante do *SFMEI*, que possui também como participante uma instituição de ensino IE_B, com uma solução com a mesma finalidade de melhoria do consumo de energia, entretanto que utiliza Raspberry Pi para implementar os dispositivos locais. Por meio da abstração fornecida pelo *SFMEI* pretende-se prover uma interface simplificada para que ambas as aplicações, implementadas em plataformas diferentes, possam fazer uso e contribuir com os modelos gerenciados pelo *SFMEI*.

O *FedIoT* define uma arquitetura para aplicação de AF em IoT baseada em três camadas principais: Aplicação; Algoritmo e Infraestrutura, conforme ilustrado na Figura 8. Esta organização demonstra um esquema básico padrão que pode ser adaptado e utilizado em diver-

sos contextos, uma vez que possui um bom nível de detalhamento e uma alta capacidade de abrangência.

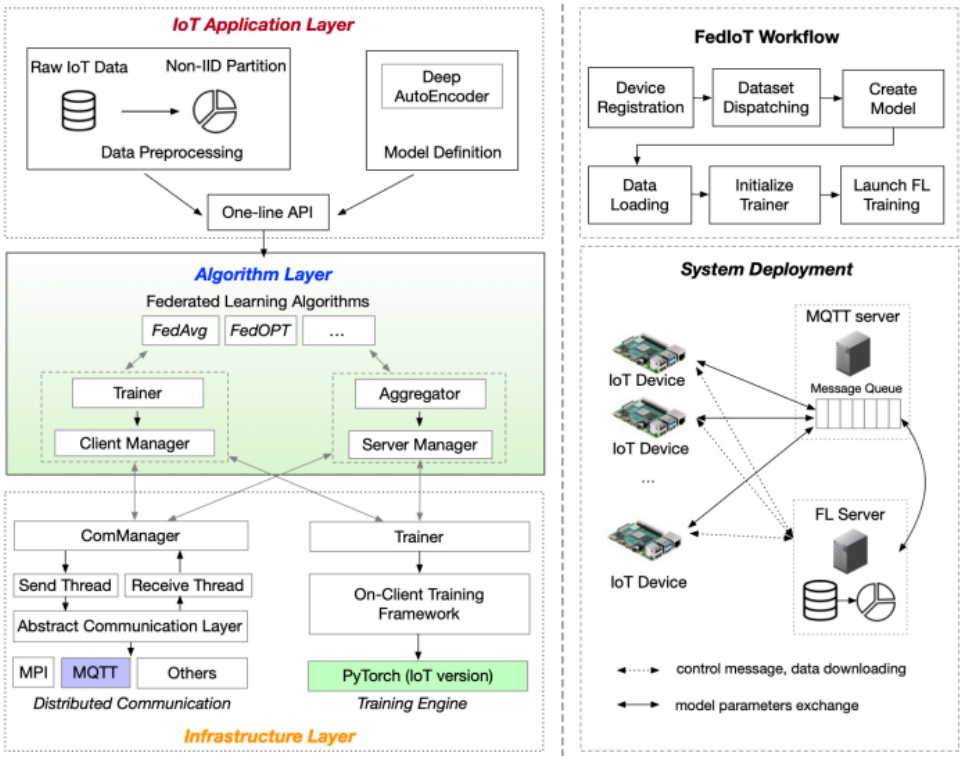


Figura 8 – Visão geral da arquitetura do *FedIoT*

Fonte: Adaptado de (Zhang et al., 2021b)

As três camadas são descritas em mais detalhes a seguir:

> "Na camada de aplicação, o FedIoT fornece uma API *one-line* (tomando o modelo *Autoencoder* e o *data loader* como entradas) para iniciar o treinamento federado em dispositivos IoT de maneira a utilizar uma computação distribuída; Na camada de algoritmo, o FedIoT suporta FedDetect e outras linhas de base algorítmicas, como FedAvg[2], FedOPT[3]; Na camada de infraestrutura, o FedIoT visa oferecer suporte a APIs de comunicação leves com MQTT [4] (ou seja, Transporte de Telemetria de Enfileiramento de Mensagens [5], um padrão para mensagens IoT) e biblioteca PyTorch personalizada para plataformas de IoT [6], que podem dar suporte ao treinamento *on-device* para dispositivos de borda IoT habilitados para CPU ou GPU, como Raspberry Pi e NVIDIA Jetson Nano." (Zhang et al., 2021b).

Para atender melhor aos objetivos deste trabalho, optou-se por desenvolver um *middleware* próprio, baseando-se na arquitetura definida por (Zhang et al., 2021b), em vez de utilizar

[2] MCMAHAN, Brendan et al. Communication-efficient learning of deep networks from decentralized data. In: **Artificial intelligence and statistics**. PMLR, 2017. p. 1273-1282.

[3] REDDI, Sashank et al. Adaptive federated optimization. **arXiv preprint arXiv:2003.00295**, 2020.

[4] HUNKELER, Urs; TRUONG, Hong Linh; STANFORD-CLARK, Andy. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In: **2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)**. IEEE, 2008. p. 791-798.

[5] *Message Queuing Telemetry Transport*

[6] PASZKE, Adam et al. Pytorch: An imperative style, high-performance deep learning library. **Advances in neural information processing systems**, v. 32, p. 8026-8037, 2019.

o FedIoT de maneira direta. Essa decisão foi baseada em uma análise comparativa entre as capacidades do FedIoT e as necessidades específicas para alcançar o objetivo deste trabalho. Por meio das lacunas e limitações no FedIoT, observadas na Tabela 4, foi possível evidenciar que a solução de um *middleware* próprio mostrou-se uma abordagem para o alcance dos objetivos.

Tabela 4 – Comparação entre SFMEI e FedIoT

Comparações	**SFMEI**	**FedIoT**
Capaz de treinar modelos através de aprendizado federado	✓	✓
Implementa estrutura de filas para comunicar com nós	✓	✓
Possui interface de acesso simplificada	✓	✓
Possui base de dados para armazenamento de modelos treinados	✓	
Possui histórico de execuções	✓	
Possibilita que aplicação interna do nó não interfira no treinamento	✓	
Possibilita ao nó final definir se participa ou não do treinamento	✓	
Possibilita ao nó final consumir modelos pré-treinados	✓	

De maneira distinta do FedIoT, o *SFMEI* possibilita uma estrutura adicional, a qual suporta uma infraestrutura de armazenamento.

3 METODOLOGIA

Quanto à metodologia aplicada neste trabalho, foi utilizada uma adaptação do método *Design Science Research* (DSR), seguindo outros estudos e pesquisas relacionados a *smart campus* (Dresch; Lacerda; Junior, 2020; Blazevic; Riehle, 2023), que também adaptaram esta metodologia proposta por (Peffers et al., 2007). Amplamente adotada, a DSR tem a seguinte definição, por (Pimentel; Filippo; Santoro, 2019):

> Na DSR, o pesquisador está comprometido com dois objetivos: (1) resolver um problema prático num contexto específico por meio de um artefato e (2) gerar novo conhecimento científico. Portanto, dois ciclos de pesquisa estão inter-relacionados na DSR: um sobre o projeto do artefato, denominado Ciclo de Design (HEVNER, 2007)[1] ou Ciclo de Engenharia (WIERINGA, 2014)[2], cujo objetivo é projetar um artefato para solucionar um problema real em um determinado contexto; e outro denominado Ciclo de Conhecimento ou Ciclo do Rigor, sobre a elaboração de conjecturas teóricas relacionadas ao comportamento humano ou organizacional.

Assim, optou-se em seguir o DSR pois, por meio da criação de uma nova solução, ou aprimoramento de uma já existente, é possível gerar um conhecimento aplicável e útil para a solução de problemas.

Quanto à classificação, este trabalho pode ter sua metodologia de pesquisa detalhada conforme os seguintes elementos:

- Natureza → Aplicada, que objetiva gerar conhecimentos para aplicação prática solucionando problemas específicos;
- Abordagem → Quantitativa, que objetiva analisar estatisticamente os dados coletados por meio do *middleware* proposto;
- Tipo → Empírica descritiva, que descreve as características do *middleware* proposto;
- Procedimentos técnicos → pesquisa bibliográfica e análise de dados.

As etapas do processo metodológico foram divididas da seguinte forma:

1. Revisão bibliográfica;
2. Mapeamento da solução;
3. Mapeamento de dados de avaliação;

[1] HEVNER, Alan R. ***A three cycle view of design science research***. Scandinavian journal of information systems, v. 19, n. 2, p. 4, 2007.

[2] WIERINGA, Roel J. ***Design science methodology for information systems and software engineering.*** Springer, 2014.

4. Mapeamento de métricas;
5. Construção do *middleware*;
6. Teste preliminares do *middleware*;
7. Avaliação do *middleware*.

Estas etapas podem ser acompanhadas detalhadamente nas seções a seguir.

3.1 Revisão de Bibliográfica

Uma revisão bibliográfica foi realizada com o intuito de aprimorar o entendimento e conhecimento acerca do cenário sendo estudado. Para isso, optou-se por realizar uma busca por trabalhos, publicados no período entre 2021 e 2023, relacionados ao cenário de *smart campus*, a fim de entender o estado da arte do tema de estudo, em vez de produzir uma revisão sistemática de literatura. Assim, foi possível deter mais atenção durante o ciclo de engenharia, como parte do DSR. Esta escolha se deu também frente à existência de algumas revisões sistemáticas recentes encontradas, descritas na Seção 2.4. O trabalho descrito em (Chagnon-Lessard et al., 2021) realizou uma revisão de uma década, entre os anos de 2011 e 2021, e detalhou, dentro de sete categorias, trinta e sete outras sub-categorias, com grande detalhe de técnico e tecnológico. Optou-se por seguir esta categorização devido a esta especificidade relacionada ao detalhamento técnico e tecnológico, que difere das outras revisões avaliadas, que não abordam de maneira tão detalhada seus domínios e dimensões (Zhang et al., 2022; Nóbrega; Chim-Miki; Castillo-Palacio, 2022; Giuriatti et al., 2023).

3.2 Mapeamento da Solução

O *framework* proposto por (Chagnon-Lessard et al., 2021) serviu como base para auxiliar o mapeamento da solução abordada nesta dissertação. Na Tabela 3, mostrada no Capítulo 2, são listadas diversas propostas de *frameworks* que implementaram aprendizado federado, ligados a soluções *smart campus* ou não. Conforme as categorias levantadas por (Chagnon-Lessard et al., 2021), foi realizada a categorização dos estudos da Tabela 3, que permitiu evidenciar a não existência de uma proposta específica de um *middleware* que se correlacione de maneira abrangente com outras categorias no desenvolvimento de soluções para *smart campus*, como pode ser visto no Apêndice B.

Uma vez identificada a proposição de um *middleware*, percebeu-se a possibilidade de habilitar um serviço, que proporcione o fácil acoplamento de instituições de ensino a soluções de Inteligência Artificial por meio de uma solução de aprendizado federado. Assim, foi possível optar entre algumas abordagens de implementação, sendo as duas principais:

- **Abordagem 1**. Desenvolver uma solução, e a partir dela observar seu comportamento e viabilidade;
- **Abordagem 2**. Adaptar uma solução existente e analisar seu comportamento.

Optou-se por seguir a abordagem 2, pela possibilidade de redução do tempo necessário para construir todos os principais conceitos arquiteturais e de comunicação e viabilizar a solução final, o que distingue-se da abordagem 1, pois seria necessário implementar e avaliar distintas opções e conceitos arquiteturais, bem como seus distintos pontos positivos e negativos. Com isso, seguiu-se uma análise das propostas de *frameworks* levantados na Tabela 3. Com um intuito de buscar uma estrutura abrangente, que suportasse sua adaptação ao cenário de um *smart campus* e, juntamente a isso, permitisse incluir detalhes de camadas de interação com o usuário, a fim de abstrair o processo do algoritmo de aprendizado federado. Desta maneira, optou-se por seguir a proposta do *framework* de aprendizado federado para detecção de anomalias em dispositivos IoT, desenvolvido por (Zhang et al., 2021b).

3.2.1 Arquitetura

A arquitetura do *framework* descrito em (Zhang et al., 2021b) permite aplicar modularizações e, onde seria assumido um nó como um dispositivo IoT, que comunicaria com um serviço central através de uma API online, foi substituído pela idealização de instituições de ensino participantes do *middleware*. Esta adaptação foi necessária pela proposta de que a arquitetura de componentes e serviços pudessem ser utilizados de forma abstraída, implementando a complexidade no *middleware* e em um cliente local, presente na instituição de ensino. Assim, o *framework* descrito em (Zhang et al., 2021b) foi adaptado para que, em vez de refletir uma estratégia de aprendizado federado *cross-device*, pudesse seguir uma estratégia *cross-silo*, por melhor adaptar-se à proposta de aprendizado federado a um cenário de um *smart campus* considerado nesta dissertação, onde, conceitualmente, o *smart campus* seria detentor e controlador dos dados obtidos pelos seu sensores locais e sua aplicação local itermediaria o processo de aprendizagem federada, como também o consumo de modelos pré-treinados.

3.3 Mapeamento dos Dados de Avaliação

Para validação de uma versão de protótipo do *middleware*, foi utilizado o conjunto de dados aberto *Hourly Energy Consumption - Over 10 years of hourly energy consumption data from PJM in Megawatts* (a ser aplicado no cenário abordado posteriormente na Seção 3.6.1), com dados pertencente a PJM Interconnection LLC[3]. Os dados deste conjunto referem-se ao consumo estimado de energia em *Megawatts* (MW) por hora, no período entre 1998 e 2018.

[3] **PJM** - Organização regional dos Estados Unidos que opera uma rede de transmissão de energia elétrica, composta por diversas companhias - https://www.pjm.com/about-pjm

Foram utilizados os dados das variáveis DAYTON[4], DOM[5], AEP[6] e PJME (PJM *East Region*, que o conjunto compreende as companhias presentes no eixo leste da rede da PJM). Por meio deste conjunto buscou-se emular aplicações de medição e predição de consumo de energia em um *smart campus*.

Para uma validação seguinte da versão final do *middleware* (cenário abordado posteriormente na Seção 3.6.2) foi utilizado o conjunto de dados ***Smart Campus Oulu indoor climate, air-quality and motion***, como conjunto de dados principal, utilizando variáveis de emissão de CO_2 (Eldeeb; Alves, 2023; Oulu, 2023). Juntamente a ele, os seguintes conjuntos de dados complementares também foram utilizados:

- ***ODDs: Occupancy Detection Dataset***. Utilizando métricas de temperaturas medidas em moradias (Makonin et al., 2016; Makonin, 2015);
- ***AMPds2: The Almanac of Minutely Power dataset (Version 2)***. Utilizando métricas de consumo de água (Weng; Zhang; Xia, 2018; Makonin, 2016);
- ***UNICON: An Open Dataset of Electricity, Gas and Water Consumption in a Large Multi-Campus University Setting***. Utilizando métricas de consumo de energia (Moraliyage et al., 2022; Moraliyage et al., 2018).

Todos esses conjuntos de dados foram coletados e definidos a partir da existência de medidas e registros em séries temporais, o que possibilita o treinamento e avaliação de modelos de regressão.

3.3.1 Tratamento dos conjuntos de dados

Cada conjunto de dados foi dividido em 5 partes para compreender os dados referentes a 1 servidor (*aggregator*) e 4 clientes (*workers*), identificados como: (i) *dataset_server.csv*; (ii) *dataset_client_1.csv*; (iii) *dataset_client_2.csv*; (iv) *dataset_client_3.csv* e (v) *dataset_client_4.csv*.

Os *scripts* criados para tratamento de cada um dos conjuntos de dados podem ser conferidos no Apêndice E.

Não foi aplicada a nenhum desses dados tratados alguma estratégia direta de normalização, remoção de *outliers*, ou aplicado algum algoritmo de interpolação, ou KNN, como aplicado por (Eldeeb; Alves, 2023), de forma a tentar não influenciar resultados melhores na execução do *middleware*.

Para cada conjunto de dados foi dado um nome de referência para um modelo. Na Tabela 5 estão descritos os nomes utilizados como referência para os modelos gerados nos cenários de teste a partir de cada conjunto de dados.

[4] **DP&L Inc.** - https://www.aes-ohio.com
[5] **Dominion Energy, Inc.** - https://www.dominionenergy.com
[6] **American Electric Power** - https://aep.com

Tabela 5 – Nomes dos modelos utilizados como referência para cada conjunto de dados.

Conjunto de dados	**Variável**	**Nome do modelo**
Smart Campus Oulu indoor climate, air-quality and motion	CO_2	*model co2*
ODDs: Occupancy Detection Dataset	Temperatura	*model temperature*
AMPds2: The Almanac of Minutely Power dataset (Version 2)	Consumo de energia elétrica	*model energy consumption*
UNICON: An Open Dataset of Electricity, Gas and Water Consumption in a Large Multi-Campus University Setting	Consumo de água	*model water*

3.3.2 Características dos dados

Os dados destes conjuntos foram tratados e organizados para seguir um esquema de dados em duas colunas: (i) *"instant"* e (ii) *"data"*. Este tratamento se deu para possibilitar o treinamento de um modelo de regressão, representando em uma variável explicativa, um instante de tempo, e uma variável resposta, um determinado dado medido no instante. (Bruce; Bruce, 2019; Harrison, 2019).

Na Tabela 6 estão detalhadas as características das colunas resultantes dos dados tratados, seus tipos e sua descrição. A coluna *"instant"* possui um tipo de dado "*timestamp*", que representa numericamente em milissegundos um instante no tempo. Já a coluna *"data"* representa como "string" a medida de um dado numérico. Optou-se por manter essa representação como "string" pois estes dados serão armazenados em uma tabela de contexto de uma base de dados a ser consumida internamente pelo *SFMEI*. Com isso mantém-se uma padronização para as tabelas de contexto, evitando que sejam utilizados tipos numéricos distintos para armazenar os dados localmente em cada cliente. Internamente no *SFMEI* os dados dessa coluna são transformados em um tipo numérico padronizado, para assim seguir o treinamento do modelo de regressão.

Tabela 6 – Detalhamento dos tipos de dados tratados.

Coluna	**Tipo**	**Descrição**
instant	*timestamp*	Instante de leitura do dado coletado
data	*string*	Medida/dado efetivamente coletado

3.3.3 Características do modelo preditivo

Um modelo de regressão simples foi escolhido de forma a se distinguir do estudo descrito em (Eldeeb; Alves, 2023), que aplica um modelo de classificação. Busca-se também contribuir com estudos mais recentes que analisem modelos de regressão em *smart campus* como por exemplo em (Netto et al., 2022).

Ademais um modelo de regressão é tipicamente utilizado em previsão de séries temporais, além de possibilitarem uma simples interpretação, facilitando a análise. Outro aspecto é a flexibilidade, podendo ser adaptados para diferentes tipos de dados, tendo em vista as distintas variáveis e métricas dos distintos conjuntos de dados utilizados neste trabalho (Strecht et al., 2015).

Neste caso optou-se por utilizar um modelo *Long Short-Term Memory* (LSTM) em uma estrutura em cinco camadas, em sequência, para realizar a predição destas series temporais. Na Figura 9 observa-se os detalhes destas camadas, a primeira camada LSTM, com 64 unidades e função de ativação tangente hiperbólica, processa a entrada com formato (S,1), onde S representa o tamanho da sequência. Essa camada retorna a sequência completa de saídas para a próxima camada. Uma camada de *dropout* com taxa de 20% é aplicada para prevenir sobreajuste. A segunda camada LSTM, com 32 unidades, processa a saída da primeira camada e retorna apenas a saída do último passo temporal. Outra camada de *dropout* com taxa de 20% é aplicada antes da camada densa final, que possui uma única unidade e é utilizada para realizar a previsão, tipicamente em problemas de regressão. As camadas deste modelo LSTM possuem 64 e 32 neurônios (unidades), respectivamente, além de 1 neurônio na última camada densa. Assim, o total de neurônios utilizados neste modelo LSTM é de 97.

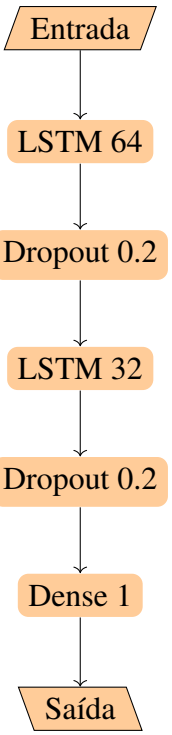


Figura 9 – Camadas LSTM *SFMEI*

Fonte: Autoria Própria

Por mim o horizonte de previsão deste modelo é de um único passo (*One-Step Forecast*). Este modelo prevê o valor do próximo ponto de tempo na série temporal. A sua representação matemática simplificada pode ser dada como:

$$\hat{y}_{t+1} = y_t \tag{1}$$

- Onde:
  - $\hat{y}_{t+1}$ valor previsto no tempo t+1;
  - y_t valor observado no tempo t

3.4 Mapeamento de Métricas

Devido a este trabalho resultar em modelos de regressão simples, foram levantadas métricas comumente utilizadas para avaliar estes modelos. Para avaliação da qualidade da predição de modelos de regressão são comumente utilizadas métricas como o escore R^2, o erro padrão residual (*Residual Standart Error* - RSE), o erro quadrático médio (*Mean Squared Error* - MSE), a raíz do erro quadrático médio (*Root Mean Squared Error* - RMSE), o erro médio absoluto (*Mean Absolute Error* - MAE), ou o erro percentual médio absoluto simétrico (*Symmetric Mean Absolute Percentage Error* - SMAPE) (Chicco; Warrens; Jurman, 2021; James et al., 2013). Cada uma destas métricas possui características específicas que auxiliam na avaliação, entretanto duas características principais podem distingui-las, as métricas RSE, MSE, RMSE e MAE possuem valores em medidas de unidade, para cada caso aplicado, já a métrica SMAPE possui valores em medida percentual. Quanto ao R^2, o mesmo possui valores pertencentes ao intervalo $\in (-\infty, 1]$, contudo, devido aos valores negativos não expressarem informações de interesse, apenas o intervalo 0 e 1 foi considerado, de modo que possam ser também inferidos como medidas percentuais.

Para este trabalho medidas percentuais possuem vantagens de utilização e interpretação, em relação a medidas de unidades de valores. Devido a utilização de variáveis com distintas unidades de valor entre os dados utilizados para avaliação (CO_2, temperatura, consumo de água e consumo de energia), não seria tão claro determinar bons valores de RSE, MSE, RMSE e MAE, quando comparados os resultados destas variáveis. Já medidas percentuais, podem expressar um valor comparativo melhor para avaliar os modelos resultantes destas distintas variáveis (James et al., 2013). Assim, optou-se por utilizar neste trabalho uma métrica percentual como métrica principal, e uma métrica de um valor de unidade como métrica segundária, de suporte.

Ao buscar estudos relativos ao desenvolvimento de *smart campus* que utilizassem medidas de métricas percentuais, como R^2 e SMAPE, para avaliar modelos de regressão, foi encontrado o estudo realizado por (Netto et al., 2022). Em (Netto et al., 2022), aplicado em outro conjunto de dados, de consumo de energia elétrica, foi possível observar que foi obtido um valor de $R^2 = 0.93$ para a avaliação de um modelo de regressão. Nele foi possível enxergar que essa métrica relaciona-se de maneira simples para avaliar conjuntos de dados de séries temporais

que não possuam uma classificação explicita. Assim, a medida do R^2 presente em (Netto et al., 2022) foi utilizada como referencial de ponto de partida como um métrica principal.

Para o caso que estamos abordando, a medida do R^2 se mostrou bastante aplicável, baseando-se também na afirmação contida em (Nakagawa; Schielzeth, 2013):

> Como R^2 não tem unidade, é extremamente útil como índice sumarizado para modelos estatísticos porque é possível avaliar objetivamente o ajuste dos modelos e comparar os valores de R^2 entre estudos de maneira semelhante às estatísticas padronizadas.

O que está em total acordo com (Chicco; Warrens; Jurman, 2021), que demonstra que o escore R^2 é uma métrica que auxilia melhor na interpretação de modelos de regressão. Por estes motivo foi dado prioridade em utilizar o R^2 como principal métrica de análise, que possui a seguinte notação matemática:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

- Onde:
  - n é o número de observações;
  - y_i são os valores observados;
  - $\hat{y}_i$ são os valores previstos;
  - $\bar{y}$ é a média dos valores observados

O escore R^2 é uma medida do percentual da variância dos dados, que pode ser utilizada para referenciar o poder de predição em relação a determinada amostra (Rigdon, 2012; Hair et al., 2019). A variância determina a distância entre o valor de uma amostra em relação à média de valores de uma predição, em que busca-se sempre o menor valor entre essa distância (Hassan et al., 2016). O escore R^2 expressa, em medida percentual, essa variância, buscando determinar o quão aproximados são os valores obtidos comparados com os valores de referência.

O R^2 possui valor entre 0 e 1, em que quanto mais próximo de 1, mais explicativo é o modelo em relação aos dados previstos (Nakagawa; Schielzeth, 2013). Entretanto, valores R^2 extremamente altos tendem a significar um sobre-ajuste(*overfitting*), isto é, o modelo gerado possui um desempenho muito bom em determinada amostra de dados, entretanto, em outras amostras seu desempenho pode ser muito ruim e não suficiente (Hair et al., 2019).

Contudo, apenas a análise do valor do escore R^2 de maneira isolada pode não ser suficiente para avaliar um modelo. Embora o R^2 indique a capacidade do ajuste do modelo, pode não avaliar diretamente a precisão das previsões. Ou seja, um modelo pode possuir possuir uma capacidade de ajuste alta frente a determinado conjunto de dados, mas resultar em uma precisão baixa de previsão, ao utilizar outra métrica em outro conjunto de dados, com já abordado no caso de um modelo em sobre-ajuste, por exemplo (Cohen et al., 2013). O mesmo ocorre para parte

dos casos, onde o valor de R^2 não demonstra-se como tão alto, próximo a 1, onde deve-se sempre utilizar outras métricas secundárias como comumente utilizado o erro quadrático médio (*Mean Squared Error* - MSE) e a raiz do erro médio quadrático (*Root Mean Squared Error* - RMSE), de forma a contribuir para uma visão mais abrangente sobre o desempenho do modelo (Cohen et al., 2013; Nakagawa; Schielzeth, 2013).

Com isso, seguindo as medidas e análises do estudo realizado no principal conjunto de dados tomado como referência (Eldeeb; Alves, 2023), pode-se avaliar a função de perda, resultante do treinamento do modelo. Uma função de perda é utilizada para avaliar e quantificar a diferença entre as previsões feitas pelo modelo e os valores reais dos dados de treinamento. Através da redução desta função, no decorrer do treinamento, é possível ao modelo aprimorar suas previsões progressivamente (Bishop; Nasrabadi, 2006). A função de perda auxilia também a identificar "o quão otimizados são os pesos, relacionados com os dados de testes previamente conhecidos, sendo uma das formas de medida mais popular para séries temporais" (Eldeeb; Alves, 2023).

De maneira a contribuir de maneira complementar com a análise dos modelo gerados, optou-se por utilizar a métrica do MSE para a avaliação função de perda, por ser a mais comum para avaliar modelos de regressão (James et al., 2013; Eldeeb; Alves, 2023). Para o MSE, quanto mais alto o valor calculado, maior o erro, que é determinado a partir da diferença entre o valor observado e o valor na predição. Essa diferença é elevada ao quadrado, para que assim valores positivos e negativos não se anulem (Hassan et al., 2016; Harrison, 2019; Bruce; Bruce, 2019).

Além dela, as métricas MAE e RSME também serão utilizadas para, por fim, compor um conjunto de métricas comumente utilizadas para avaliar modelos de regressão (Chicco; Warrens; Jurman, 2021). Por fim, as métricas utilizadas neste trabalho para avaliar os modelos gerados pelo *SFMEI* e consequentemente seu desempenho geral são: R^2, MAE, MSE, e RSME.

3.5 Construção do *Middleware*

Foi montado um ambiente de simulação que permitisse realizar o treinamento de modelos de maneira distribuída entre distintos clientes, enquanto o servidor pudesse aplicar algoritmo de federação para agregar estes modelos. Para isso, foi desenvolvido e estruturado o *SFMEI* através da linguagem de programação *Python* [7]. Foram desenvolvidas duas aplicações, servidor e cliente, utilizando o *framework Flask* [8], que possibilita que sejam disponibilizadas APIs REST (*Representational State Transfer*). A API do servidor, pensada para ser privada (não disponibilizada ao cliente final), foi criada pensando em permitir a consulta de dados de modelos pré-treinados no repositório do servidor e gestão de recursos e utilização. Já a API do cliente permite que sejam realizadas consultas ao repositório local, e criadas e alteradas configurações de execução, além de abstrair consultas de dados de modelos pré-treinados à API do servidor.

[7] ***Python*** - www.python.org
[8] ***Flask*** - flask.palletsprojects.com

Foram desenvolvidos conectores para fazer uso de um serviço de filas e tópicos. Foi experimentado o uso de *MQTT* para este caso, funcionando de maneira simples (Santos et al., 2022). Entretanto, visando melhor acompanhamento das filas criadas, juntamente à possibilidade de gestão por interface de usuário visual, optou-se por ter um *Broker* de comunicação utilizando *RabbitMQ*[9]. O *RabbitMQ* foi também utilizado por possibilitar que sejam criados serviços de fila, retirando assim a responsabilidade e necessidade de constante conexão entre o *publisher* e o *subscriber*. Através de uma estrutura de filas é possível que o *SFMEI* trabalhe com um acoplamento menor entre as aplicações de cliente e servidor. Com isso, se diferencia do *framework* FedIoT, proposto por (Zhang et al., 2021b), que implementa sua comunicação através de *MQTT*, em uma estrutura de publicação/assinatura(*publish/subscriber*), o que precisa garantir que o tópico e conexão seja mantida por ambas as pontas.

Uma vez possuindo todo o controle de comunicação, através das APIs REST do cliente e do servidor, juntamente com o *broker* de comunicação, em seguida foi desenvolvida a lógica que possibilitasse o treinamento de modelos. Assim, primeiro desenvolvido o serviço de agregação, presente no servidor do *SFMEI*, de modo que fosse possível gerar um modelo global inicial e em seguida o distribuir para os clientes. Em seguida, partiu-se para o *SFMEI Local Client*, que recebe e parametriza o modelo, carrega seus dados locais para treinamento, e por fim devolve este modelo ao servidor do *SFMEI*. Ambos os serviços de aprendizado, no *SFMEI* e *SFMEI Local Client*, foram desenvolvidos utilizando o *framework TensorFlow*[10].

Por fim, foi criada e definida toda a lógica que possibilita o armazenamento de modelos em repositórios no servidor *SFMEI* e *SFMEI Local Client*. Desta forma, foi estruturado toda o esquema de base de dados, que utiliza um banco de dados *MariaDB*[11]. Essa escolha se deu por dois fatores principais, o primeiro que o *MariaDB* é *open source* e totalmente licenciado sob GPL (*General Public License*), e o segundo, que possui mais mecanismos de armazenamento e melhor desempenho, quando comparado com *MySQL*[12], por exemplo (AWS, 2023). No *SFMEI* essa característica se traduziu na possibilitar de utilizar de forma nativa uma coluna que armazena dados do tipo 'json' e poder consultá-lo de forma mais rápida.

Embora não explorados, bancos de dados *NoSQL*, que não utilizam o modelo relacional tradicional baseado em tabelas, também podem ser uma opção possível de solução para o *SFMEI*, principalmente para lidar diretamente com as informações dos modelos treinados. Entretanto, por incluir uma estrutura básica relacional para gestão de conexões e clientes, optou-se por seguir apenas com uma instância simples de um único banco de dados, o *MariaDB*.

Toda essa estrutura de simulação foi montada e viabilizada através de *containers* Docker [13] a fim de emular um cenário onde "maquinas" distintas possuísem contextos isolados,

[9] ***RabbitMQ*** - www.rabbitmq.com
[10] ***Tensorflow*** - www.tensorflow.org
[11] ***MariaDB*** - mariadb.com
[12] ***MySQL*** - www.mysql.com
[13] ***Docker*** - www.docker.com

mantendo apenas as comunicações via API e Broker entre si. As imagens utilizadas para os serviços foram:

- **python:3.9-slim**[14] - Imagem utilizada para serviços das aplicações do servidor *SFMEI* e *SFMEI Local Client*;
- **rabbitmq:3-management**[15] - Imagem utilizada para serviço do *Broker* de comunicação;
- **mariadb:10.5.8**[16] - Imagem utilizada para serviço do banco de dados do servidor *SFMEI* e do *SFMEI Local Client*.

Os arquivos **.yml** de configuração **Docker compose** e os arquivos **Dockerfile**, utilizados para construir e instanciar os serviços *Docker*, estão detalhados no Apêndice C.

3.6 Teste do *middleware*

Foram idealizados alguns cenários de testes com a finalidade de validar e avaliar o *SFMEI* em diversos aspectos, sendo:

- Cenário de teste 1 - Avaliação comparativa entre o *SFMEI* e abordagem de aprendizado de máquina não federado.
- Cenário de teste 2 - Avaliação comparativa entre algoritmos de agregação de aprendizagem federada FedAVG e FedSGD.
- Cenário de teste 3 - Avaliação do desempenho do *SFMEI* provendo modelos pré-treinados.

3.6.1 Cenário de teste 1 - Avaliação comparativa entre o *SFMEI* e abordagem de aprendizado de máquina não federado

Neste cenário de teste foi idealizada uma comparação de uma aplicação para predição em um cenário não federado e um protótipo do *SFMEI*, que utiliza aprendizagem federada. A comparação realizada neste cenário tem o intuito probatório de validar se há, ou não, prejuízos quanto a capacidade de predição, ao utilizar o *SFMEI*. Busca-se neste cenário validar se *SFMEI* é capaz de alcançar métricas similares ao treinamento de um modelo em uma abordagem de aprendizagem de máquina não federada enquanto mantém os ganhos e pontos positivos da aprendizagem federada, dentre eles, principalmente o não compartilhamento de dados locais, utilizados para treinamento.

[14] ***python:3.9-slim*** - hub.docker.com/layers/library/python/3.9-slim/images/sha256-54b1eb6dee478a3105c78fecfe75d796963537b76cc6fb7fffd54d2f70695543

[15] ***rabbitmq:3-management*** - hub.docker.com/layers/library/rabbitmq/3-management/images/sha256-55155e28c2fd4741067ba8e905d6e753bea5914a3a646d40d04c9b7cc2a8fa5a

[16] ***mariadb:10.5.8*** - hub.docker.com/layers/library/mariadb/10.5.8/images/sha256-03fb19fa5729856ec8c8ed23d421ed1ab6c0e2d63fdf2b1bd8d311025e228a9b

Na aplicação não federada o cenário contemplou o treinamento de um modelo durante 1000 épocas, definido arbitrariamente como uma condição de parada. De maneira a equivaler, no protótipo do *SFMEI*, cada nó treinou o modelo por 100 épocas, durante 10 rodadas de aprendizagem federada, (mantendo assim uma "pseudo-equivalência" de treinamento de 1000 épocas em cada conjunto de dados). O treinamento foi realizando utilizando os dados das companhias DAYTON, DOM e AEP, resultando em um o modelo a ser avaliado com os dados da companhia PJME.

Este cenário busca predizer valores de uma série temporal contínua, de uma variável em MW, ao longo do tempo (James et al., 2013). Assim, foi decidido por realizar o treinamento de um modelo de regressão, utilizando *Long Short-Time Memory* (LSTM), comumente utilizado para predizer cenários de séries temporais a vários anos, como por ser visto nos estudos realizados em (Fushiki, 2011), (Wong; Yeh, 2019), (Shah et al., 2022) e também em (Zhang; Liu, 2023), por exemplo, e que possui uma boa performance para realizar estas predições, como visto em (Bolboacă; Haller, 2023).

3.6.2 Cenário de teste 2 - Avaliação comparativa entre algoritmos de agregação de aprendizagem federada FedAVG e FedSGD

Frente a existência de distintos algoritmos de agregação de aprendizagem federada, descritos na Seção 2.2.1, foram idealizados neste cenário dois casos de teste que distinguem-se entre si, sendo (i) Caso de teste A - Algoritmo FedAVG e (ii) Caso de teste B - Algoritmo FedSGD. Ambos os cenários são composto por um servidor central e quatro clientes distribuídos. Todos os quatro clientes possuem dados locais relacionados a contextos de "Emissão de CO_2", "Temperatura" e "Consumo de Energia Elétrica". Além disso, dois destes quatro clientes possuem dados relacionados a "Consumo de Água", para contribuir com uma visão complementar de uma análise do desempenho do treinamento com um número reduzido de modelos. Buscou-se neste cenário avaliar para qual algoritmo de agregação o *SFMEI* possui melhor desempenho, utilizando como métrica o valor calculado do R^2, juntamente com o valor da função de perda, pela métrica MSE.

Em ambos os cenários foi utilizado um modelo *Long Short-Time Memory* para treinamento. Em nenhum dos cenários foi aplicado um limite de rodadas de aprendizagem. Foi utilizado um limitador quanto ao valor calculado do R^2. Caso o valor de avaliação do modelo atingisse um valor maior ou igual a 0.99 não seriam mais executadas rodadas de federação para esse modelo, pois este valor de R^2 indicaria que este modelo pode ter atingido um sobreajuste, de acordo com o classificado por (Hair et al., 2019). Sendo esta a única condição de parada adotada.

Entretanto, durante a execução de cada algoritmo, o *SFMEI* manteve-se em execução ininterrupta durante um intervalo maior que 25 horas, em seguida foi interrompido. Desta maneira tomou-se esse valor como valor de corte para análises, definido de modo arbitrário, por não haver referencial para casos de aprendizagem contínua, como o *SFMEI*, nos estudos de referência,

conforme Tabela 3.

3.6.3 Cenário de teste 3 - Avaliação do desempenho do *SFMEI* provendo modelos pré-treinados

Este cenário foi idealizado a ser aplicado ao caso de teste do cenário anterior que possuir melhor desempenho, utilizando ele em uma simulação do consumo de modelos pré-treinados. A execução dentre os algoritmos FedAVG e FedSGD (casos de teste A e B do cenário de teste 2) que obtiver melhor desempenho será definida para ser utilizada em uma simulação em que o *SFMEI* possui modelos pré-treinados em seu repositório e clientes que não participaram do treinamento destes modelos consumirão o melhor modelo para realizar predições locais.

Esta simulação é composta de três clientes hipotéticos X, Y e Z. Estes clientes não participam do treinamento de modelos, mas consomem os modelos pré-treinados por meio da interface pública do *SFMEI Client*, destinada ao uso por parte destes clientes. Os três clientes possuem dados locais distintos, que não fizeram parte em nenhum momento do treinamento de modelos. Estes dados serão utilizados para avaliar um cenário de predição, simulando uma comparação com dados "reais".

Busca-se com esse cenário realizar uma avaliação mais detalhada da capacidade de predição de um modelo pré-treinado disponibilizado. Cada cliente hipotético consumirá o modelo melhor avaliado e utilizará dos locais, não utilizados para treinamento do modelo, os dados resultantes da predição serão comparados com os dados locais através das métricas R^2, MAE, MSE, e RSME.

3.7 Avaliação do *Middleware*

Com base no trabalho descrito em (Hair et al., 2019), é possível classificar um modelo de acordo com o valor do seu R^2. Na Tabela 7 está detalhada essa classificação. A execução do *middleware* proposto visa classificar os escores R^2 resultantes conforme a categorização de (Hair et al., 2019), amplamente adotada como referência para a classificação de modelos com base no valor do R^2.

Tabela 7 – Categorização no modelo baseado no R^2.

Classificação	**Valor do R^2**
Fraco	< 0.50
Moderado	< 0.75
Suficiente	> 0.75
Sobreajuste	$>= 0.99$

Juntamente ao R^2, o MSE possibilita contribuir para a análise e avaliação de modelos que possam obter valores de R^2 não tão altos. Por meio da visualização da funcão de perda ao longo do tempo, é possível observar, tanto em uma amostra de teste quanto em outra de validação, uma

diminuição do MSE. Desta forma, é possível identificar caso ocorra uma aproximação de zero, do valor do MSE calculado, ou um afastamento do zero. A aproximação de zero significa uma tendência à convergência por parte do modelo, mesmo que ele tenha um valor de R^2 baixo.

Estas métricas serão utilizadas a fim de comparar o desempenho do *SFMEI* utilizando os algoritmos de FedAVG e FedSGD. Por fim, serão analisados os modelos pré-treinados melhor classificados pelo algoritmo que possuir melhor desempenho dentre os dois. Com isso, será avaliado e determinado o desempenho geral do *SFMEI*, utilizando o conjunto de métricas definidas (R^2, MAE, MSE, e RSME), aplicadas em uma simulação na qual clientes consomem os modelos pré-treinados por esse algoritmo.

4 *MIDDLEWARE* SFMEI

Neste capítulo é apresentada a visão geral do *middleware* proposto, denominado *Smart Federated Middleware for Educational Institutions (SFMEI)*, que possui como finalidade simplificar o desenvolvimento e a adaptação de aplicações *IoT* de *Smart Campus* que usam aprendizado de máquina federado ao prover uma interface de abstração dos componentes do *middleware*.

4.1 *Smart Federated Middleware for Educational Institutions*

O *Smart Federated Middleware for Educational Institutions (SFMEI)* surgiu com o intuito de oferecer uma camada de abstração para possibilitar que aplicações *IoT* em *smart campus*, que usam Inteligencia Artificial, possuam acesso simplificado a modelos de aprendizado de máquina enquanto contribuem de maneira federada, sem ferir a privacidade. Com isso, o *SFMEI* foi idealizado para que, utilizando um cliente local, o *SFMEI Client*, seja possível que uma instituição de ensino se conecte ao *middleware* para utilizar seus serviços, contribuindo para o desenvolvimento e transformação dos ambientes educacionais em ambientes inteligentes.

Uma vez conectada ao *SFMEI*, a instituição pode ter acesso a uma interface pública, que possibilita os dois cenários de usos:

- Cenário 1: Colaborar de maneira participativa, utilizando seus dados locais para treinamento de modelos locais que posteriormente podem participar das rodadas de maneira federada
- Cenário 2: Fazer uso de um repositório de modelos pré-treinados, que possibilitar a geração de predições de cenários locais, ainda que não tenha participado do treinamento dos modelos.

Além destes cenários, o *SFMEI* é capaz de atender aos requisitos de privacidade, relacionados aos dados confidenciais acerca dos ambientes educacionais, possibilitando o desenvolvimento de soluções inteligentes para estes ambientes de maneira segura.

4.2 Arquitetura da Solução Proposta

A arquitetura do *SFMEI* distribui-se na seguinte estrutura organizacional: (i) Aplicação; (ii) Algoritmo; (iii) Infraestrutura e (iv) Armazenamento; que, em um alto nível, caracterizam-se da seguinte maneira:

- **Aplicação**. Estrutura funcional, que possibilita a utilização do *SFMEI*. Provê um meio de acesso tanto para inserir dados gerados por dispositivos IoT como para a obtenção de

informações acerca de modelos treinados. Este meio de acesso permite que os modelos pré-treinados possam ser utilizados para gerar predições e também que os modelos gerados localmente possam ser trafegados entre os domínios da arquitetura. Através do uso de uma Interface de Programação de Aplicação (*Application Programming Interface* - API) é possível atender a estas demandas, abstraindo a lógica interna do *SFMEI* e detalhes dos algoritmos de aprendizagem federada;

- **Algoritmo**. Abrange os algoritmos de agregação são responsáveis por intermediar o relacionamento entre clientes (*workers*) e o servidor (*aggregator*), sendo capaz de gerenciar a agregação de modelos globais, a partir dos modelos treinados localmente pelos clientes. Nela está a responsabilidade de avaliar o desempenho dos modelos treinados, a fim de alcançar algum valor máximo de qualidade determinada pelo agregador. Por exemplo, se um agregador definir que o mínimo aceitável é que um modelo global possua um escore $R^2 = 0,96$, as rodadas de federação deverão continuar ocorrendo enquanto o modelo global possua escores inferiores. Nesta organização, possibilita que sejam implementados distintos algoritmos de agregação;

- **Infraestrutura**. - Quanto a infraestrutura, a proposta do *SFMEI* oferece uma estrutura mínima de recursos necessários para viabilizar a comunicação e gestão de todos os módulos do componentes. Por meio de um comunicador principal, é possível que a abstração e adaptabilidade sejam mantidas, de modo que os recursos estruturam-se para possibilitar o treinamento de modelos locais no cliente e a agregação de modelos no servidor central. Juntamente abrange os recursos necessários para armazenamento dos dados e modelos treinados pelo *SFMEI*;

- **Armazenamento**. - Compreende os repositórios pertencentes aos clientes e servidor. Abrange a estrutura de tabelas responsável por armazenar os conjuntos de dados de contexto, utilizados para treinamento de modelos locais no cliente e para validação e avaliação de modelos no servidor. Ademais também gere o armazenamento de configurações e informações principais para funcionamento do *SFMEI*.

Conforme descrito na Seção 2.4.3, o sistema de implantação do FedIoT (Figura 8) se demonstra como sendo uma opção simples, mas que não descreve, tampouco aborda as interfaces de uso, nem possibilitam que os clientes optem por participar dos treinamentos ou não. Entretanto, este mesmo sistema de implantação e comunicação demonstra-se como uma opção de rápida adaptação para o cenário proposto para a atuação do *SFMEI*. Baseando-se nele foi possível adaptar e evoluir esta estrutura de comunicação e interação, a fim de possibilitar troca de informações entre o *SFMEI* e as instituições de ensino.

O *SFMEI* difere-se também do FedIoT, ao disponibilizar filas específicas para cada cliente conectado ao servidor. Esta estrutura possibilita que cada cliente possa optar por participar do aprendizado federado treinando apenas modelos específicos, pois irá consumir apenas os modelos

comunicados e enviados para suas filas. O que não é possível através do FedIoT, pois todos os clientes terão de manter-se sempre conectados e inscritos e uma única fila de mensagem.

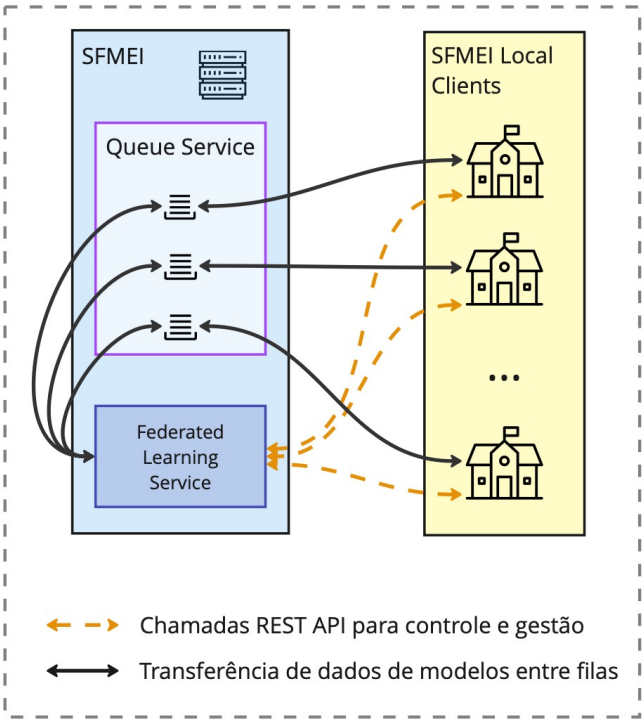


Figura 10 – Sistema de implantação

Fonte: Autoria Própria

A estrutura mostrada na Figura 10 detalha o sistema de implantação inicial, o qual possibilitou chegar a arquitetura ilustrada na Figura 11. Em ambas as imagens pode-se observar que a implementação do *SFMEI* abrange dois domínios principais: (i) Domínio do servidor do *middleware*, em azul; (ii) Domínio da instituição de ensino, em amarelo.

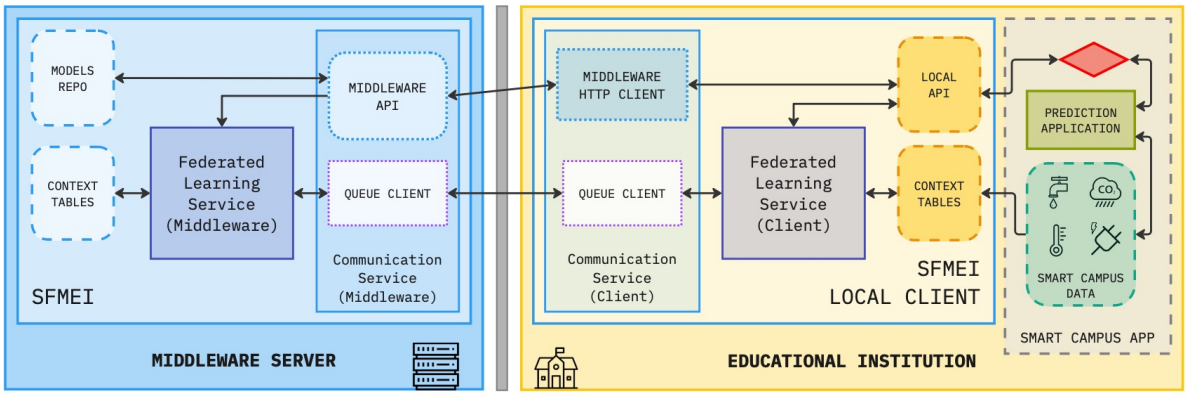


Figura 11 – SFMEI - Arquitetura

Fonte: Autoria Própria

Nota-se que toda a estrutura baseia-se em uma aplicação central no domínio do servidor e uma aplicação local no domínio da instituição de ensino. A aplicação do servidor leva o

nome do *middleware*, *SFMEI*, já a aplicação local é uma extensão chamada de *SFMEI Local Client*. A fronteira entre ambos os domínios é transpassada através de dois meios principais de comunicação. Uma API REST, que gere o tráfego de informações principais e o serviço de filas RabbitMQ, que realiza a gestão de tópicos de dados e eventos das aplicações.

4.3 *SFMEI* - Aplicação

Neste conjunto estão dispostas e implementadas as principais responsabilidades de do *SFMEI*: (i) fornecer uma interface ao usuário final; (ii) interagir com as demais componentes do sistema; (iii) implementar a lógica da aplicação; (iv) gerenciar a persistência de dados gerados nas mais diversas fontes de armazenamento.

4.3.1 Interfaces

No contexto do *SFMEI*, a camada de interfaces desempenha o importante papel de proporcionar o acesso tanto aos dados originados nos clientes quanto às informações provenientes dos modelos treinados. A presença de um meio de acesso nessa camada possibilita a utilização de modelos pré-treinados para gerar predições, ao mesmo tempo em que facilita o tráfego de modelos gerados localmente entre os diferentes domínios da arquitetura. Dessa forma, a utilização de uma API se mostra como uma solução eficaz, que visa atender às demandas do *SFMEI*, garantindo que ocorra uma comunicação direta em todo o sistema.

Uma API RESTful, comumente chamada de API REST, é baseada em um conjunto de princípios, que definem como os recursos são representados e tratados em uma rede, sendo principalmente baseada em HTTP. Além das APIs REST, existem outros tipos de APIs (Gough; Bryant; Auburn, 2021), como:

- APIs SOAP (*Simple Object Access Protocol*): APIs baseadas em XML que utilizam o protocolo SOAP para comunicar com um servidor;
- APIs GraphQL: APIs que permitem que os clientes solicitem apenas os dados que precisam. Elas utilizam um protocolo de consulta declarativa para especificar os dados que desejam recuperar.

Neste trabalho, optou-se por utilizar APIs REST nas interfaces públicas e privadas do *SFMEI*, com base em (Gough; Bryant; Auburn, 2021), que recomenda essa abordagem pela simplicidade e flexibilidade que oferece. Ademais o uso de métodos HTTP facilita a integração e adaptação do *SFMEI* às soluções locais nas instituições de ensino.

4.3.1.1 Interface pública

A API pública do *SFMEI Local Client* opera como um acesso intermediário que abstrai a lógica interna do *SFMEI*, bem como os detalhes específicos dos algoritmos de aprendizagem

federada. Essa abstração torna simplificada a implementação de funcionalidades e aplicações presentes no domínio do cliente, ou seja, uma instituição de ensino. Essa API possui dois tipos distintos de endereços (*endpoints*). Os *endpoints* locais, que implicam diretamente na gestão de modelos e dados localmente e os *endpoints* de acesso, utilizados para intermediar consultas ao repositório do servidor do *SFMEI*.

Na Tabela 8 pode-se observar os principais *endpoints* locais que fazem parte do *SFMEI Local Client*.

Tabela 8 – *Endpoints* locais do *SFMEI Local Client*

Método HTTP	**Endpoint**	**Descrição**
POST	/model/train/run	Adiciona a configuração de um modelo candidato a treinamento, caso não exista. Ou, caso exista, altera a configuração deste modelo para torná-lo apto ao treinamento.
POST	/model/train/stop	Altera a configuração do modelo para tornar um determinado modelo não apto a treinamento. Isso resulta que o modelo não será considerado para treinamento em nenhuma rodada de aprendizagem federada.
POST	/model/update	Adiciona novos dados manualmente na tabela de contexto.

Na Tabela 9 estão detalhados os *endpoints* locais que fazem parte do *SFMEI Local Client* e que são utilizados para realizar consultas ao repositório do *SFMEI*.

Tabela 9 – *Endpoints* para consultas ao repositório do *SFMEI*

Método HTTP	**Endpoint**	**Descrição**
GET	/categories	Lista todos as categorias registradas no *SFMEI*.
POST	/category	Busca no repositório do *SFMEI* o detalhe de uma categoria, por nome, contendo a lista de todos os modelos treinados dessa categoria.
POST	/model/get	Busca um modelo pré-treinado por nome e categoria.

4.3.1.2 Interface Interna

Já a API interna e privada do *SFMEI* visa mitigar riscos de exposição indevida, garantindo a integridade do processo de comunicação e aprendizagem. Estes aspectos são possíveis através de uma padronização que simplifica e facilita a interação entre *SFMEI* e *SFMEI Local Client*.

Na Tabela 10 estão detalhados os *endpoints* que provém a intermediação entre *SFMEI Local Client* e *SFMEI*, possibilitando que sejam realizadas consultas aos recursos presentes no *SFMEI*, como, por exemplo, o repositório de modelos pré-treinados.

Tabela 10 – *Endpoints* internos que intermedeiam consultas ao repositório do *SFMEI*

Método HTTP	**Endpoint**	**Descrição**
GET	/middleware/categories	Lista todos as categorias registradas no *SFMEI*.
POST	/middleware/category	Busca no repositório do *SFMEI* o detalhe de uma categoria, por nome, contendo a lista de todos os modelos treinados dessa categoria.
POST	/middleware/model/get	Busca um modelo pré-treinado por nome e categoria.

Na Tabela 11 detalham-se os *endpoints* de controle do *SFMEI*. Estes, que são endereços que possibilitam registro e conexão entre os clientes *SFMEI Local Client* e *SFMEI*.

Tabela 11 – *Endpoints* de controle (registro e conexão) entre o *SFMEI Local Client* e *SFMEI*

Método HTTP	**Endpoint**	**Descrição**
GET	/middleware/ready	Verifica se o *SFMEI* está online e pronto para receber modelos para treinamento.
POST	/middleware/register	Registra clientes no middleware e retorna um identificador único de cliente e as configurações do broker de comunicação.
POST	/middleware/connect	Conecta um cliente registrado ao *SFMEI*.

4.3.2 Cenários

Nesta camada do *SFMEI* também está presente toda a estrutura que possibilita a interação entre as demais camadas do sistema. Com o *SFMEI Local Client*, é possível que a instituição de ensino interaja com toda a estrutura do *SFMEI* através de dois cenários de uso principais:

- **Cenário 1**. Colaborando com treinamento de modelos, no qual a instituição pode utilizar os recursos disponibilizados pelo *SFMEI* para colaborar utilizando seus dados locais para o treinamento de modelos, utilizando aprendizagem de máquina federada. Por meio do *SFMEI Local Client*, uma instâcia de um banco de dados é disponibilizada para a instituição de ensino, possibilitando uma maneira fácil de estruturar os seus dados locais a serem usados para o treinamento de modelos. Juntamente a esse recurso, a interface pública, detalhada na Seção 4.3.1.1, possibilita que seja realizada de forma simples a configuração necessária para que novos modelos sejam treinados localmente, utilizando os dados locais.

  Para participar do treinamento de modelos é necessário que, primeiramente, seja criada uma tabela de contexto com um esquema de dados no padrão descrito na Seção 3.3.1. Modelos distintos necessitam de conjunto de dados distintos, consequentemente tabelas de contextos distintas. Por exemplo, para treinar dois modelos para predição de dois contextos, como o consumo de água e consumo de energia, se faz necessária a criação de

duas tabelas de contexto distintas. Uma vez criada a tabela, deve-se agregar os dados à tabela de contexto. Por meio de uma chamada a API REST é possível apontar a tabela de contexto, incluindo o modelo local a ser treinado dentre as configurações de execução do *SFMEI*.

Na Tabela 12, estão elencadas as etapas de funcionamento, detalhadas na Figura 12, que ilustra este cenário.

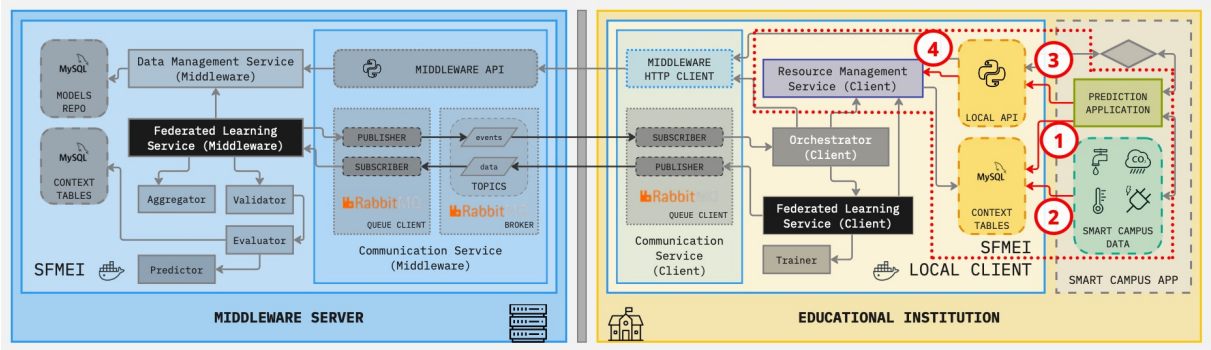


Figura 12 – Cenário em que o cliente colabora com o *SFMEI* no treinamento de modelos através de aprendizado federado

Fonte: Produzido pelos autores

Tabela 12 – Etapas cenário 1 - Colaborando com treinamento de modelos.

Etapa	Atuação	Descrição
1	Usuário	Criação de tabela de contexto.
2	Usuário	Inserção de dados na tabela de contexto.
3	Usuário	Inclusão de configuração.
4	*Middleware*	Registo de modelo candidato a treinamento.

- **Cenário 2**. Utilizando o *SFMEI* como repositório de modelos pré-treinados, no qual a instituição pode utilizar os recursos disponibilizados pelo *SFMEI* para consultar e fazer usos de modelos pré-treinados mediante o aprendizado de máquina federado. Por meio da interface pública do *SFMEI Local Client*, detalhada na Seção 4.3.1.1, a instituição de ensino é capaz de consultar e buscar os modelos pré-treinados. Estes modelos podem ser utilizados nas aplicações locais da instituição de ensino para gerar predições nos seus contextos locais.

  Por meio de uma chamada à API REST, é possível consultar no repositório do *SFMEI* quais são os modelos pré-treinados disponíveis. Assim, pode-se buscar um modelo pontualmente a fim de utilizá-lo para realizar predições dos cenários locais.

  Na Tabela 13 pode-se encontrar as etapas de funcionamento, detalhadas na Figura 13, que ilustra quando o *SFMEI* pode ser utilizado como um repositório de modelos pré-treinados.

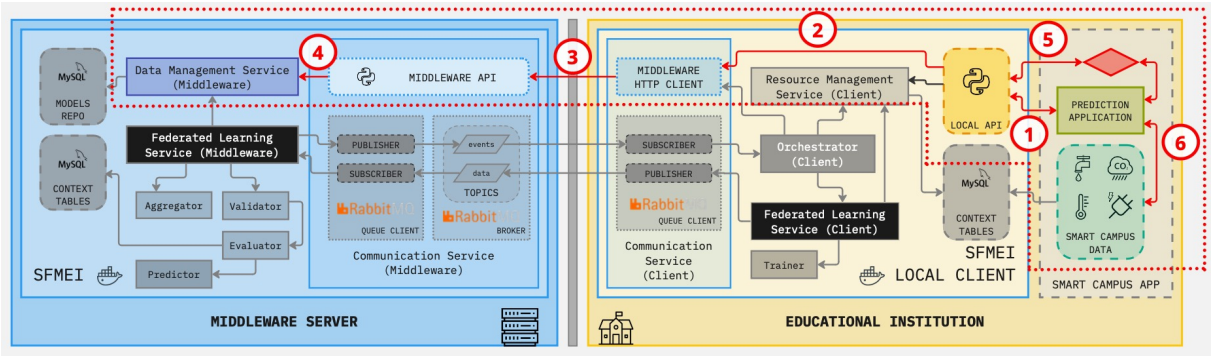


Figura 13 – Cenário em que o *SFMEI* é utilizado como repositório de modelos pré-treinados

Fonte: Produzido pelos autores

Tabela 13 – Etapas cenário 2 - Utilizando *SFMEI* como repositório de modelos pré-treinados.

Etapa	Atuação	Descrição
1	Usuário	Consulta de modelos pré-treinados disponíveis.
2	*Middleware*	Requisição a serviço interno de comunicação.
3	*Middleware*	Requisição através de chamada a API Rest privada do *SFMEI*.
4	*Middleware*	Consulta de modelos no repositório do *SFMEI*.
5	Usuário	Requisição de modelo pré-treinado.
6	Usuário	Utilização de modelo pré-treinado para predição.

4.3.3 Fluxo de funcionamento

Toda a lógica de funcionamento do *SFMEI* também está presente nesta camada. Podemos detalhá-la em dois principais fluxos: (i) o fluxo de treinamento de modelo local, no cliente, e (ii) o fluxo de agregação, no servidor.

4.3.3.1 Fluxo de treinamento de modelo local no cliente

A Figura 14 evidencia, dentro da arquitetura, o fluxo de treinamento de modelos localmente. Todo o fluxo é gerenciado a partir de um serviço orquestrador, e ocorre da seguinte maneira:

1. O fluxo inicia-se quando o orquestrador do *SFMEI Local Client* consome no tópico de eventos uma mensagem, publicada pelo *SFMEI*, informando qual o modelo possui pesos disponíveis para treinamento;

2. O orquestrador valida se o modelo recebido está marcado, dentre as configurações locais, como um modelo candidato para treinamento;

- Caso o modelo não esteja marcado como candidato o orquestrador aguarda consumir a mensagem seguinte do tópico de eventos.

3. Caso esteja marcado como candidato para treinamento, o orquestrador requisita ao serviço de comunicação local os parâmetros do modelo disponível;
   - Este serviço realiza uma chamada à interface privada requisitando os parâmetros;
   - Internamente, no *SFMEI*, é buscado no repositório do servidor o modelo e seus parâmetros.
4. Uma vez recebidos os parâmetros, em seguida é iniciado o serviço local de aprendizado federado. Este serviço carrega estes parâmetros no modelo local;
5. Em seguida, o serviço local de aprendizado federado solicita os dados da tabela de contexto do modelo;
6. Os dados de contexto são carregador e podem ser transformados e divididos em uma estrutura de treinamento e de validação;
   - Por exemplo, 75% da tabela de contexto é utilizada para treinamento dos modelos e 25% para validação;
7. O serviço local de aprendizado federado dispara um treinamento para aquele modelo;
8. Por fim, os parâmetros do modelo após treinamento e validação são publicados no tópico de dados, juntamente com as métricas de execução do modelo.

Todo o fluxo é repetido de maneira contínua enquanto houver, dentre as configurações, o modelo como candidato a processamento. No instante em que a instituição de ensino não queira mais participar do treinamento de modelos, ela possui a autonomia de encerrar, e retomar a qualquer momento. Para isso, basta ajustar as configurações do modelo candidato a treinamento.

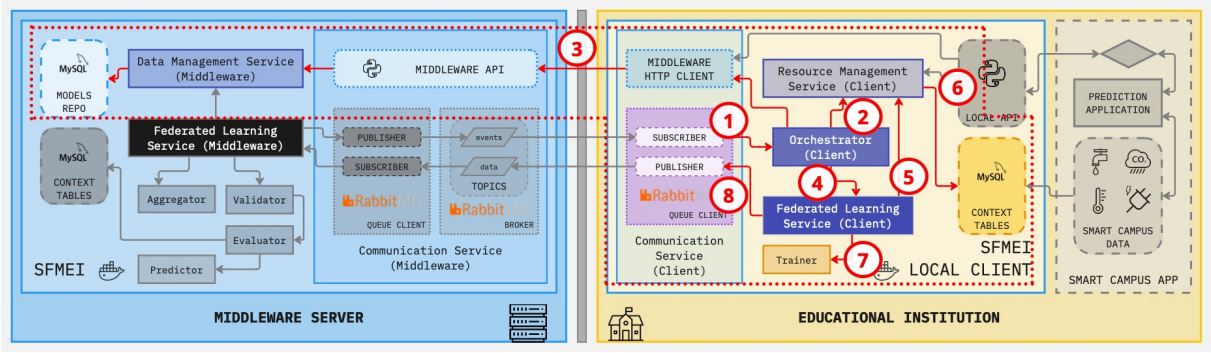


Figura 14 – Fluxo em que o modelo é treinado localmente pelo cliente com seus dados locais

Fonte: Produzido pelos autores

4.3.3.2 Fluxo de agregação no servidor

Em evidência na Figura 15 está identificado o fluxo de agregação de modelos por parte do servidor. No *SFMEI* o fluxo é gerenciado a partir de um serviço de aprendizagem federada da seguinte maneira:

1. Este serviço possui um agendamento de execução por um determinado período de tempo, por exemplo, a cada 10 minutos, que requisita ao serviço de gerenciamento de dados o próximo modelo para treinamento;
2. O serviço busca no repositório e valida dentre os modelos qual modelo não está em treinamento no momento;
3. Caso não haja treinamentos prévios, esse modelo é treinado, para que, logo em seguida, seja publicado. E, caso afirmativo, buscou-se os pesos do modelo para continuidade do fluxo
4. O modelo a ser treinado através de aprendizagem federada é publicado nos tópicos de eventos do *SFMEI*;
    - Cada conexão registrada no *SFMEI* possui, no serviço de *RabbitMQ*, um tópico de eventos e dados único.
5. Uma vez o modelo treinado pelos clientes, o serviço de aprendizagem federada consome este modelo e seus parâmetros;
6. Uma vez os parâmetros consumidos o serviço de agregação valida a regra utilizada (qual estratégia, qual algoritmo de federação) e aplica a agregação de acordo com essa regra;
7. O novo modelo global agregador é repassado para do componente de validação, que inicia o processamento final desse novo modelo;
8. Em seguida, o modelo é repassado para um serviço de avaliação;
9. Logo após são buscados os dados da tabela de contexto do *SFMEI*;
10. Estes dados da tabela de contexto são utilizados, pelo componente de avaliação para, através de uma predição, avaliar o modelo agregado;
11. Após avaliado o serviço de federação registra todas as informações na base de dados local.

Este fluxo é continuadamente repetido, mantendo sempre um modelo em toda a estrutura do *SFMEI* e *SFMEI Local Client*.

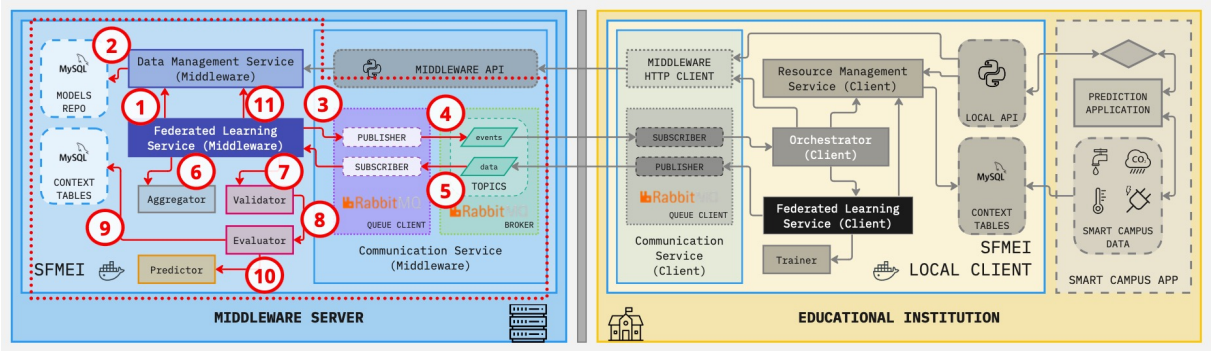


Figura 15 – Fluxo em que o modelo treinado pelo *SFMEI*, notificado para treinamento pelos clientes, em seguida, recebido, agregado e validado.

Fonte: Autoria própria

4.4 *SFMEI* - Algoritmo

Na Seção 2.2.1 foram descritos os dois principais algoritmos de aprendizado federado existentes atualmente. O FedAVG, que tem como foco a agregação através do cálculo da média dos gradientes e o FedSGD, que utiliza uma abordagem em que cada cliente atualiza o modelo global de forma independente, aprimorando o modelo melhor avaliado dentre os modelos treinados. O *SFMEI* abrange toda a implementação e gestão destes ambos os algoritmos.

4.4.1 FedAVG

Como já abordado, o FedAVG se destaca pela sua simplicidade, em que seu funcionamento básico se dá de acordo com os seguintes passos: (i) O agregador recebe os modelos de cada dispositivo; (ii) o agregador calcula a média entre os modelos recebidos; (iii) o agregador gera um novo modelo global a partir da média calculada. Essa simplicidade foi norteadora para incorporação no *SFMEI*.

A Figura 16 detalha o funcionamento do *SFMEI* quando incorporado o FedAVG. Nela podemos identificar o funcionamento da seguinte maneira:

1. O processo inicia após o consumo da notificação do tópico de dados. O *SFMEI* busca em seu repositório todos os modelos treinados por categoria;

2. Com os modelos já obtidos, o *SFMEI* calcula a média entre todos os modelos;

3. Realiza um treinamento local com seus próprios dados de validação e, em seguida, avalia o escore resultante desse modelo, verificando se ele atingiu um escore de referência pré-definido;

4. Se o escore for satisfatório, ou seja, se o modelo atingir o escore de referência (melhor escore obtido até o momento), o *SFMEI* define esse novo modelo como referencial no

melhor modelo obtido, caso não, SFMEI notifica todos os clientes através do tópico de eventos;

5. O *SFMEI Local Client* consome a notificação do tópico de eventos, treina o modelo localmente com seus dados locais e em seguida notifica os dados do modelo treinado no tópico de dados.

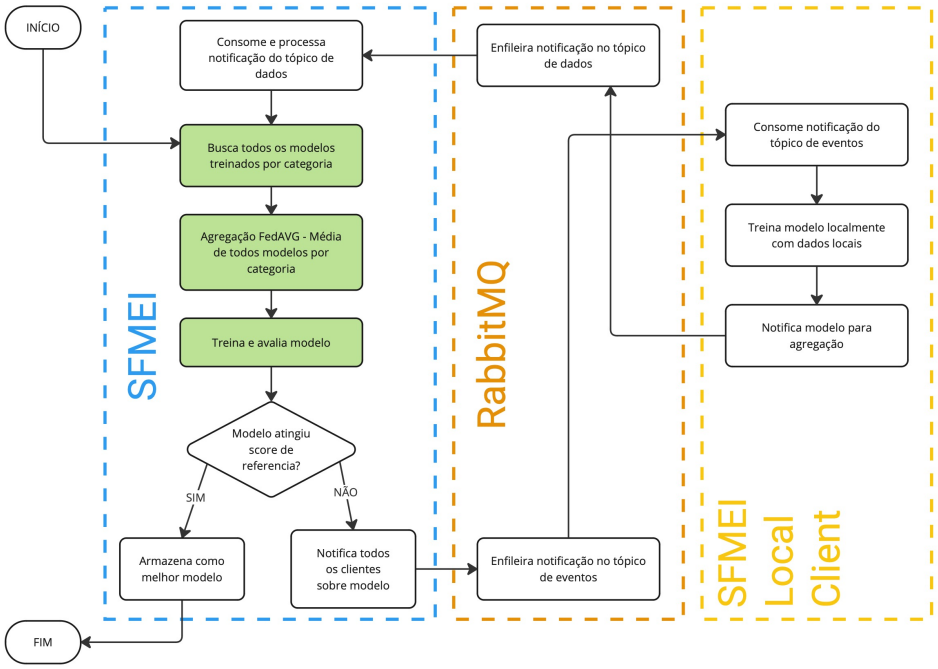


Figura 16 – Algoritmo FedAVG implementado no *SFMEI*

Fonte: Produzido pelos autores

4.4.2 FedSGD

O FedSGD se concentra em encontrar o modelo ideal entre os modelos treinados pelos clientes. O modelo escolhido é utilizado como base para o treinamento nas próximas rodadas de aprendizagem de (Kontar et al., 2021). Para isso, o algoritmo comumente segue os seguintes passos: (i) O agregador recebe os modelos de cada dispositivo; (ii) O agregador avalia cada modelo enviado e seleciona o melhor, de acordo com uma métrica específica; (iii) O agregador envia o modelo melhor avaliado de volta para os dispositivos, que o utilizam como base para o treinamento na próxima rodada.

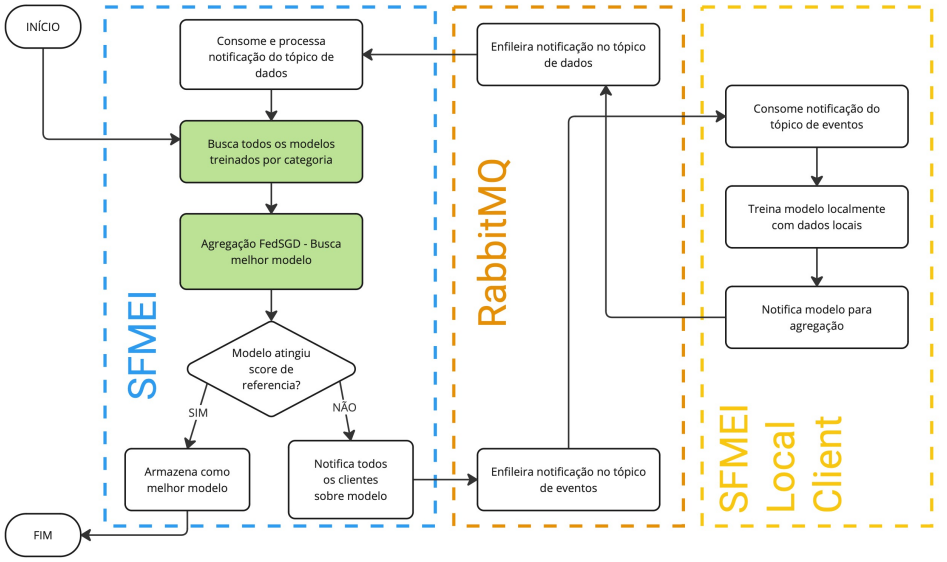


Figura 17 – Algoritmo FedSGD implementado no *SFMEI*

Fonte: Produzido pelos autores

A Figura 17 detalha o funcionamento do *SFMEI* quando implementado o FedSGD. O funcionamento deste algoritmo ocorre da seguinte maneira:

1. O processo inicia após o consumo da notificação do tópico de dados. O *SFMEI* busca em seu repositório todos os modelos treinados por categoria;
2. Com os modelos já obtidos, o *SFMEI* busca dentre eles o modelo melhor avaliado, de acordo com o escore R^2;
3. Se o escore do melhor modelo for satisfatório, ou seja, se o modelo atingir o escore de referência (melhor escore obtido até o momento), o *SFMEI* define esse novo modelo como referencial no melhor modelo obtido, caso não, o SFMEI notifica todos os clientes por meio do tópico de eventos;
4. O *SFMEI Local Client* consome a notificação do tópico de eventos, treina o modelo localmente com seus dados locais e em seguida notifica os dados do modelo treinado no tópico de dados.

4.5 *SFMEI* - Infraestrutura

Em um nível mais alto, a infraestrutura do *SFMEI* é composta de diversos componentes interligados, abrangendo servidores, a aplicação de middleware, bancos de dados, além da aplicação cliente, o *SFMEI Local Client*. Servidor e cliente, *SFMEI* e *SFMEI Local Client* respectivamente, foram implementados utilizando *containers* e imagens *Docker* para viabili-

zar e instanciar os componentes de toda essa infraestrutura. As principais imagens utilizadas foram **python:3.9-slim**, **mariadb:10.5.8** e **rabbitmq:3-management**.

4.5.1 Infraestrutura servidor e cliente

A imagem **python:3.9-slim**, baseada na distribuição **Linux Debian**, foi utilizada para a criação de *containers* que viabilizam os principais serviços dos componentes presentes no servidor *SFMEI* e cliente *SFMEI Local Client*. Esta imagem deriva-se da imagem **debian:12-slim**[1], imagem oficial da distribuição **Linux Debian 12**, conhecida por suportar múltiplas arquiteturas de hardware (Project, 2024), além de ainda comumente ser utilizada em cenários experimentais e pesquisas em IoT e *smart cities* (Lupton et al., 2022; Alhaj et al., 2023; Gore et al., 2023). Juntamente, a imagem utilizada possui apenas uma instalação básica da linguagem **Python**, versão **3.9.19** e suas dependências principais.

A utilização desta imagem proveu um ecossistema básico em **Python**, que possibilitou o uso do *framework* ***TensorFlow***, que compõe a principal biblioteca para aprendizado de máquina federado. Além disso, também possibilitou ser desenvolvida a estrutura da API responsável pelas principais conexões e comunicações entre *SFMEI* e *SFMEI Local Client*. Por meio de uma aplicação **Flask** no servidor é possível ao *SFMEI* gerenciar o repositório de modelos pré-treinados, a conexão com filas e tópicos de comunicação, a agregação de modelos recebidos dos clientes e o armazenamento de metadados dos modelos. Já no *SFMEI Local Client*, a aplicação desenvolvida em **Flask** possibilita treinar modelos localmente, enviar modelos treinados para o servidor e carregar dados locais para treinamento.

4.5.2 Infraestrutura de armazenamento

O armazenamento de dados foi possibilitado através do uso de um *container* **Docker** com a imagem **mariadb:10.5.8**, tanto para os serviços parte do servidor *SFMEI*, como do cliente *SFMEI Local Client*. Esta imagem possui uma instância do banco de dados **MariaDB**, que é um sistema de gerenciamento de banco de dados relacional. Optou-se por seu uso por derivar-se do **MySQL**, mas que, diferentemente deste, possui apenas licença de código aberto, GPL (MySQL, 2010; MariaDB, 2024). Foi descartado o uso de um banco de dados não relacional pela necessidade do *SFMEI* em garantir uma maior integridade referencial dos dados e suas estruturas.

No servidor *SFMEI*, está presente a estrutura de gerenciamento do repositório de modelos pré-treinados, o armazenamento da gestão de conexão e registro dos clientes ao servidor, bem como os metadados dos modelos recebidos e agregados durante todo o processo de aprendizagem federada. Já no cliente *SFMEI Local Client*, estão armazenados os dados de contexto para

[1] ***debian:12-slim*** - hub.docker.com/layers/library/debian/12-slim/images/sha256-993f5593466f84c9200e3e877ab5902dfc0e4a792f291c25c365dbe89833411f

treinamento dos modelos treinados localmente, os metadados dos modelos treinados, juntamente à toda estrutura de registro do cliente e também as referências de conexão ao servidor.

4.5.3 Infraestrutura de comunicação

A infraestrutura de comunicação do *SFMEI* utiliza diversas tecnologias e protocolos para garantir que os dados sejam trocados de forma confiável entre os componentes do sistema. De forma mais específica, a comunicação entre servidor *SFMEI* e cliente *SFMEI Local Client* utiliza um *Broker* de Comunicação e um conjunto de APIs. O *broker* de comunicação, implementado com **RabbitMQ**, gerencia filas e tópicos de comunicação, buscando garantir que as informações sejam trocadas de forma eficiente e segura. Já as APIs REST facilitam a comunicação entre cliente e servidor, buscando garantir uma interface padronizada de comunicação.

O *broker* de comunicação foi estruturado em um *container* **Docker** que faz uso da imagem **rabbitmq:3-management**. Essa imagem possui uma instância de uma aplicação **RabbitMQ**, que gerencia filas e tópicos de comunicação, garantindo que as informações sejam trocadas de forma eficiente e segura. O **RabbitMQ** é uma ferramenta que facilita a comunicação assíncrona entre servidor e cliente, utilizando protocolos como AMQP (*Advanced Message Queuing Protocol*). Optou-se, principalmente, pelo uso de **RabbitMQ** pela possibilidade da uma futura demanda de escalabilidade, juntamente com um grande número de clientes e mensagens, enquanto garante a entrega de mensagens mesmo em caso de falhas. Além disso, como ponto positivo também pode-se descrever a flexibilidade, pois pode-se explorar diferentes protocolos de mensagens além do AMQP, como o próprio MQTT e o STOMP (*Simple Text Oriented Messaging Protocol*).

As APIs REST, desenvolvidas em **Python** com o *framework* **Flask**, fornecem interfaces padronizadas para acesso a recursos do sistema via protocolo HTTP (*Hypertext Transfer Protocol*). Por essa razão, utiliza os métodos GET, POST, PUT e DELETE do HTTP para permitir o acesso e manipulação dos recursos. Estes recursos são identificados através de URLs. E, comumente, utiliza-se JSON ou XML para representar dados utilizados através destes recursos. Como citado anteriormente, APIs REST se adaptam bem a aplicações que tem como foco principal a simplicidade e flexibilidade (Gough; Bryant; Auburn, 2021).

No *SFMEI*, as APIs REST são principalmente responsáveis por intermediar as comunicações síncronas. Um exemplo dessa comunicação é na realização de consultas por parte do *SFMEI Local Client* ao repositório do modelos pré-treinados do *SFMEI*.

Por fim, não foi utilizado diretamente um protocolo ou algoritmo de proteção de dados na infraestrutura de comunicação do *SFMEI*. Optou-se por manter este aspecto não abordado no cenário emulado para que o principal esforço pudesse ser concentrado em compreender a viabilidade e desempenho do *SFMEI*.

4.6 *SFMEI* - Armazenamento

A estrutura de armazendamento do *SFMEI* tem como responsabilidade estruturar o armazenamento dos dados. Ela é composta por dois repositórios distintos, um no servidor, *SFMEI*, e outro no cliente, *SFMEI Local Client*.

O repositório do *SFMEI* é responsável por armazenar os dados do servidor. Ele é um banco de dados relacional que conta com as seguintes tabelas:

- Tabelas de controle/configuração: Essas tabelas armazenam informações sobre a configuração da aplicação, como os clientes, endereços e recursos (como parâmetros de funcionamento e execução);
- Histórico de execução de treinamento de modelos: Essas tabelas armazenam informações sobre os modelos de aprendizado de máquina que foram treinados através de aprendizagem federada. Estas informações incluem a data do treinamento, os pesos resultantes desse treinamento e também os resultados do treinamento, como as perdas calculadas em cada época de treinamento;
- Repositório de modelos pré-treinados: Essas tabelas armazenam modelos de aprendizado de máquina pré-treinados que podem ser utilizados pelo cliente *SFMEI Local Client*;
- Tabelas de avaliação: Essas tabelas armazenam as referências dos melhores modelos treinados/avaliados durante a aprendizagem federada;
- Tabelas de contexto: Essas tabelas armazenam dados a serem utilizados no treinamento e avaliação de modelos.

O repositório do *SFMEI Local Client* é responsável por armazenar os dados do cliente. Ele é um banco de dados relacional que conta com as seguintes tabelas:

- Tabelas de controle/configuração: Essas tabelas armazenam informações sobre a configuração da aplicação do cliente, das referencias dos modelos a serem treinados e recursos (como parâmetros de funcionamento e execução);
- Histórico de execução de treinamento de modelos: Essas tabelas armazenam informações sobre os modelos de aprendizado de máquina que foram treinados pelo cliente;
- Tabelas de contexto: Essas tabelas armazenam dados a serem utilizados no treinamento de modelos pelo cliente.

No caso do repositório do *SFMEI*, ele foi idealizado e projetado para suportar grandes volumes de dados, transações complexas e segurança. No caso do repositório do *SFMEI Local Client*, ele foi idealizado e projetado para ser rápido e fácil de usar.

Especificamente, o principal fator considerado para a utilização de bancos de dados como repositórios foi a manutenibilidade. Bancos de dados desacoplados da aplicação garantem um desempenho adequado, se comparado com arquivos, por exemplo, e também facilitam a manutenção e atualização destes repositórios.

5 AVALIAÇÃO DO SFMEI

Neste capítulo é apresentado o processo de avaliação do *middleware* desenvolvido. Ele está dividido em três seções: (i) **Análise de métricas**, descrevendo métricas e a maneira que serão aplicadas e avaliadas; (ii) **Resultados**, em um primeiro momento apenas expondo e descrevendo os resultados obtidos; (iii) **Discussão**, avaliando, de maneira mais detalhada, os resultados obtidos, correlacionando e avaliando de acordo com as métricas utilizadas.

5.1 Análise de métricas

Uma vez definidos os conjuntos de dados, foram aplicados os testes dos cenários descritos na Seção 3.6, com a finalidade de avaliar e validar o *SFMEI*, comparando os algoritmos FedAVG e FedSGD.

5.1.1 Análises quanto a classificação pelo valor do escore R^2

Com base na Tabela 7, montada a partir da categorização abordada em (Hair et al., 2019), foi idealizada a Tabela 14, a ser utilizada como principal referencial de classificações de modelos, nas Seções 5.2.3 e 5.2.4. Deste modo buscou-se classificar e analisar, se os modelos gerados pelo *SFMEI*, através do emprego de aprendizado federado, são: (i) Fracos; (ii) Moderados; (iii) Suficientes; ou são modelos em (iv) Sobreajuste.

Tabela 14 – Modelo de tabela classificativa baseada no valor do escore R^2.

Modelo	R^2	**Classificação**
Modelo - Caso A	v_A	c_A
Modelo - Caso B	v_B	c_B

Modelos considerados “Suficientes” possuem, segundo (Hair et al., 2019), um alto poder de predição, quando comparados com os modelos “Moderados”. Os modelos “Fracos” tendem a não conseguir realizar predições assertivas. Desta maneira, é possível considerar que o cenário que resulte nos modelos “Suficientes” se sobressaia frente aos outros. Já modelos em “sobreajuste” possuem um bom desempenho em determinada amostra de dados, mas um desempenho muito ruim em outras amostras (Hair et al., 2019).

5.1.2 Análises quanto a função de perda por meio do MSE

Foi analisada a função de perda dos modelos gerados, através da métrica do erro quadrático médio (MSE), com o intuito de possibilitar avaliar a evolução desta métrica no decorrer das épocas de treinamento, principalmente para modelos que possam ter sido classificados

como "moderados" pelo R^2, podendo auxiliar na análise da convergência de um modelo. Esta medida da função de perda é calculada utilizando seus dados de treinamento, e, através dela, é possível gerar uma visualização da medida obtida ao longo do tempo (épocas de treinamento), utilizando tanto uma amostra dados de teste, do treinamento, quanto uma amostra de dados de validação. Espera-se que seja observada uma redução gradual no MSE ao longo do tempo, com ele aproximando-se de zero.

Por meio da função de perda é possível analisar a convergência do modelo treinado. A convergência de um modelo refere-se a capacidade do mesmo em alcançar um estado estável, em que as previsões se tornam aproximadas a um valor fixo no decorrer do treinamento, e, em uma função de perda, por exemplo, os erros sejam minimizados durante o treinamento, mantendo uma tendência a zero (Neter et al., 1996).

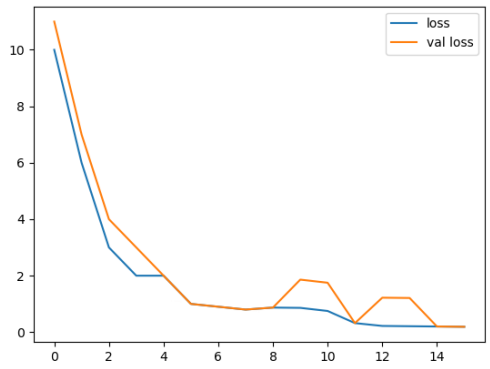


Figura 18 – Exemplo função de perda modelo convergente

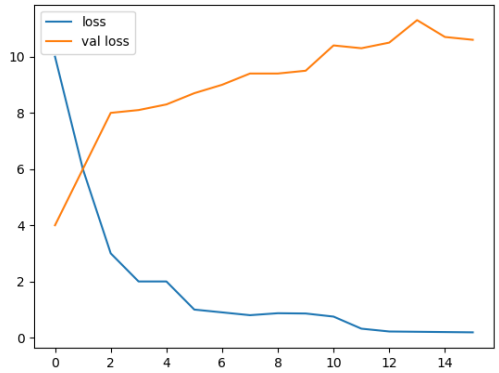


Figura 19 – Exemplo função de perda modelo não convergente

Fonte: Produzido pelos autores

Considerando um cenário hipotético, na Figura 18 é possível identificar a tendência a zero, pela função de perda, no decorrer do tempo. Esta aproximação por ambas as amostras de treinamento e de validação demonstra que o modelo treinado tende a convergir. E, possuindo MSE bastante aproximado a zero, pode realizar predições sem muita distinção do valor esperado (Rady, 2011; Shen et al., 2021).

De forma distinta, na Figura 19, é possível identificar a tendência ao distanciamento do zero, pela função de perda, ao decorrer do tempo. Este distanciamento por parte da amostra de validação demonstra que o modelo treinado tende a ser enviesado, consequentemente, não convergir, demonstrando que as predições geradas possuem muita distinção do valor esperado (Rady, 2011; Shen et al., 2021).

Neste trabalho, espera-se analisar essa função de perda utilizando o MSE. Avaliando seu comportamento através de uma validação cruzada, na proporção 25%/75%. Ou seja, 25% dos dados utilizados para treinamento serão separados para estimar a performance dos modelos treinados ao longo das épocas, tanto no servidor, quanto nos clientes.

5.1.3 Análises quanto a todo o conjunto de métricas

Além do R^2 e MSE, outras métricas serão também utilizadas para avaliar o desempenho geral do *SFMEI*. A métricas MAE e RMSE auxiliarão na avaliação o *SFMEI* em uma simulação, descrita na Seção 3.6. Como exposto anteriormente, a medida do R^2 é expressa em um valor entre 0 e 1, onde o valor mais alto, mais aproximado de 1, significa um melhor potencial preditivo do modelo avaliado. De maneira distinta as métricas MSE, MAE e RMSE possuem uma relação onde o menor valor significa uma melhor avaliação. Busca-se, com essas métricas avaliar o desempenho preditivo do *SFMEI* ao realizar predições em um cenário local em clientes que consumirão um modelo pré-treinado do *SFMEI*. Assim, através do R^2 será classificado o desempenho do modelo. As demais métricas irão demonstrar concordância quanto ao desempenho.

5.2 Resultados

Nesta seção estão expostos e apresentados os principais dados obtidos neste trabalho. Compõem esses dados resultante da avaliação dos cenários de teste descritos detalhadamente na seção 3.6). De modo a não comprometer a legibilidade do trabalho, optou-se por manter em Apêndices alguns dados brutos.

5.2.1 Resultados do cenário de teste 1 - Avaliação comparativa entre o SFMEI e abordagem de aprendizado de máquina não federado

Inicialmente, como um primeiro cenário de testes, com a finalidade de identificar se há, ou não, prejuízos ao utilizar o *middleware* proposto, um protótipo do *SFMEI* foi construído para que fosse possível compará-lo a uma abordagem de aprendizagem de máquina não federada. Um dos aspectos principais para compreender a viabilidade da proposta do *SFMEI* era entender se seria possível, utilizando o mesmo, aplicando aprendizagem federada, alcançar ao menos um potencial de predição de uma abordagem de aprendizagem de máquina não federada. Por isso, uma aplicação de aprendizagem de máquina convencional, não federada, foi implementada para ser utilizada como aplicação de referência. Além dela foi construído um protótipo do *SFMEI*, capaz de distribuir o treinamento de um modelo em 3 trabalhadores e um agregador, que seria capaz de utilizar um algoritmo de agregação FedAVG para agregar os modelos finais. Com isso, buscou-se identificar a viabilidade de implantação do *SFMEI*, se com seu uso, seria possível manter seus aspectos positivos, principalmente a privacidade dos nós, e ainda assim atingir resultados similares a abordagens de aprendizagem de máquina não federada, entendendo que, se não fosse possível alcançar um potencial de predição similiar, haveria um prejuízo em utilizar o *SFMEI*, visto a robustez de sua infraestrutura e implementação, quando comparado a uma aplicação convencional.

Foi utilizado um conjunto de dados de consumo de energia elétrica para treinamento de

um modelo de regressão. Este conjunto de dados, abertos, possui variáveis do consumo estimado de energia em *Megawatts* (MW), no período entre 1998 e 2018, e com ele buscou-se emular aplicações de medição e predição de consumo de energia em um *smart campus*. Com estes dados foi treinado um modelo LSTM para a predição do consumo de energia, tanto numa aprendizagem de máquina não federada, quanto no protótipo do *SFMEI*.

Foram utilizadas em ambas as aplicação uma condição de parada considerando o número de épocas de treinamento realizadas. Na aplicação não federada foram realizadas 1000 épocas de treinamento, onde o modelo de regressão resultante foi avaliado obteve um valor de escore $R^2 = 0,9895$. Este valor foi utilizado como valor de referência para avaliar o protótipo do *SFMEI*, que, por sua vez, em 100 rodadas de federação, no decorrer de 10 épocas de treinamento em cada cliente e servidor, pôde obter um valor de escore $R^2 = 0,9898$, conforme pode ser observado na Figura 20. Nela é possível observar que os modelos gerados foram avaliados durante cada uma das 100 rodadas de aprendizagem federada, e que, tanto os trabalhadores, quanto o agregador obtiveram resultados que se aprimoraram no decorrer das rodadas. Evidenciando também que o modelo agregado foi capaz de alcançar, o valor de referência, o que, por si só seria capaz de demonstrar que através do uso do *SFMEI* não haveriam prejuízos, quanto ao potencial preditivo do modelo gerado.

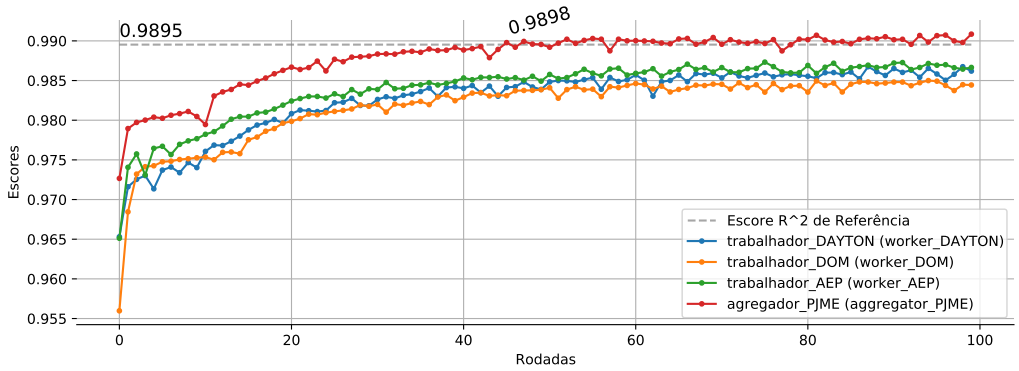


Figura 20 – Escore R^2 - Comparação entre a abordagem de aprendizagem federada do protótipo do *SFMEI* e a abordagem não federada.

Fonte: Adaptado de (Santos et al., 2022)

Alguns aspectos complementares podem ser analisados. Nesta figura é possível também evidenciar um aspecto claro do impacto do algoritmo de agregação FedAVG, Seção 4.4.1, em que observa-se que os trabalhadores resultam em valores de R^2 similares, mas que, de maneira distinta, o agregador é capaz de obter um valor de R^2 melhor em todos os casos. Isto se dá pelo fato do agregador realizar um média dos modelos treinados pelos trabalhadores, o que resulta em um modelo melhor adaptado e que possui consequentemente um maior potencial de predição. Além disso, o modelo agregador alcançou, e superou, o valor de referência em torno da rodada ± 45, o equivalente a ± 450 rodadas de treinamento, visto que em cada rodada de federação foram realizadas 10 épocas de treinamento, inclusive no agregador.

Estes resultados corroboraram para a continuidade do desenvolvimento do *SFMEI* sendo possível assim prosseguir com os cenários de teste seguintes.

5.2.2 Cenário de teste 2 - Avaliação comparativa entre algoritmos de agregação, FedAVG e FedSGD

A execução do cenário de teste 2 resultou em mais de 2200 modelos de regressão, treinados em ambos os algoritmos de agregação (FedAVG e FedSGD). Estes dados referem-se tanto aos modelos finais agregados pelo servidor do *SFMEI*, quanto pelos modelos locais treinados pelos clientes do *SFMEI*. Os dados obtidos e armazenados através do treinamento destes modelos possibilitam ir além de uma análise do desempenho geral do *SFMEI*, com eles é possível também realizar uma análise detalhada do comportamento e métricas do treinamento em cada cliente participante das rodadas de aprendizagem federada e seus desempenhos em conjunto, diferenciando mais ainda este estudo dos demais estudos e soluções de aprendizado federado encontrados até o presente momento, abordados na Seção 2.4.2 e detalhados na Tabela 3.

Tabela 15 – Sumarização dos modelos gerados

Modelo	**FedAVG**	**FedSGD**
model co2	404	283
model energy consumption	383	273
model temperature	363	271
model water	134	94
Total	1283	920

Na Tabela 15 está detalhada a quantidade de modelos treinados, distinguido por nome e algoritmo de agregação. Mesmo que a execução para cada algoritmo tenha sido interrompida em uma duração de tempo similar (25 horas), observa-se que para o algoritmo FedAVG, a execução foi capaz de gerar mais modelos que o algoritmo FedSGD no mesmo período. Desta maneira pode-se considerar que, aparentemente, através do algoritmo FedAVG, o *SFMEI* e seus clientes possuem um tempo de geração e treinamento de modelos menor, quando comparado ao algoritmo FedSGD, embora este trabalho não tenha se atido a compreender e explorar mais este aspecto, devido a seus objetivos, estes pontos podem ser melhor investigados e explorados em possíveis trabalhos futuro, entretanto se faz necessário compreender se a "agilidade" do *SFMEI* ao utilizar o algoritmo FedAVG se traduz também em um bom desempenho quanto a geração de modelos, ou seja, se utilizando o mesmo é capaz de se obter modelos com um bom potencial de predição.

5.2.3 Resultados cenário de teste 2 - FedAVG

Em uma primeira análise dos resultados obtidos neste cenário, ao expôr de maneira gráfica a quantidade de modelos gerados por classificação na Figura 21, foi possível notar que uma quantidade expressiva dos modelos gerados foram classificados como "fraco". Juntamente,

de maneira sumarizada na Tabela 16, pode-se observar que apenas ±35% dos modelos gerados pelo *SFMEI* podem ser classificados como “suficientes”, com uma mediana de escore $R^2 = 0,84$.

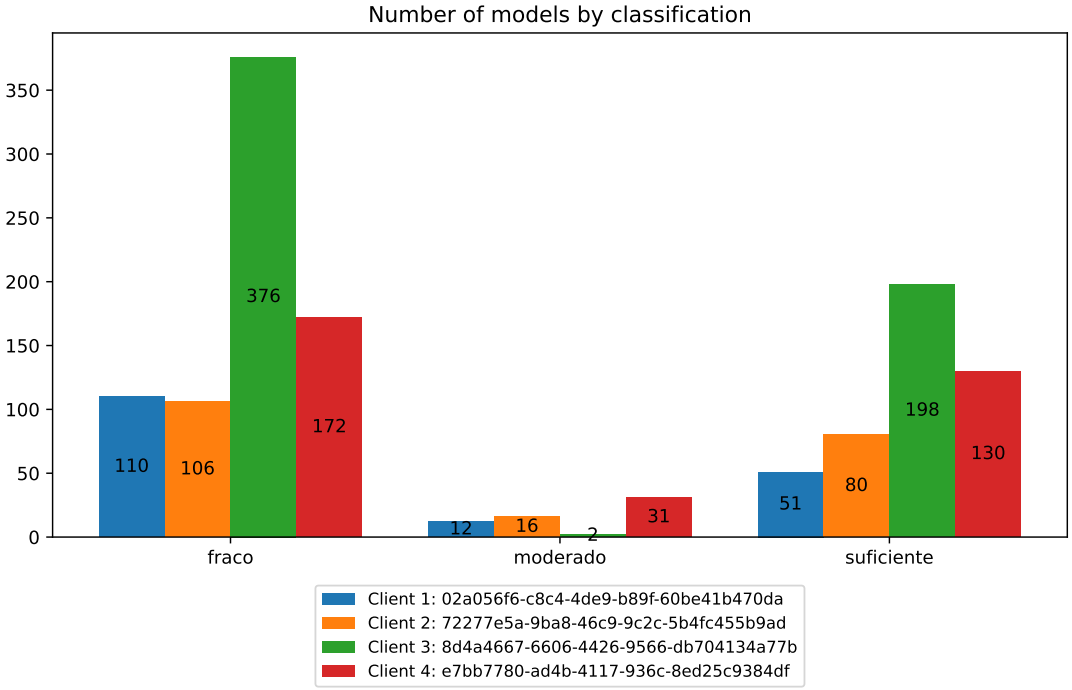


Figura 21 – Número de modelos por classificação - Treinados por todos os clientes - Algoritmo FedAVG

Fonte: Produzido pelos autores

Tabela 16 – Distribuição de frequências de modelos gerados - FedAVG

Classificação	**Quantidade**	**Percentual Classificação**	**Mediana R^2**
Fraco	764	59,50%	0.229034
Moderado	61	04,75%	0.625317
Suficiente	459	35,74%	0.846697

Analisando a Figura 22 é possível identificar melhor outros aspectos deste resultado, dentre eles, os baixos valores resultantes dos modelos “fracos” e “moderados” e também a consistência dos modelos “suficientes”. Mediante a distribuição dos escores R^2 por classificação no gráfico, pode-se observar que os modelos classificados como "fracos", acima, apresentam uma concentração significativa de valores abaixo do limiar para a classificação "moderado", evidenciado através do primeiro e segundo quartil (mediana) que estão posicionados em um valor de ±0.25. Os modelos "moderados", ao centro, apresentam uma dispersão maior dos valores de R^2, onde variam de maneira bastante distribuída entre 0.50 e 0.75, sugerindo um desempenho preditivo variável destes modelos. É possível também observar que parte dos modelos chegam a aproximar-se do limiar dos modelos “suficientes”, visto que o terceiro quartil está posicionado

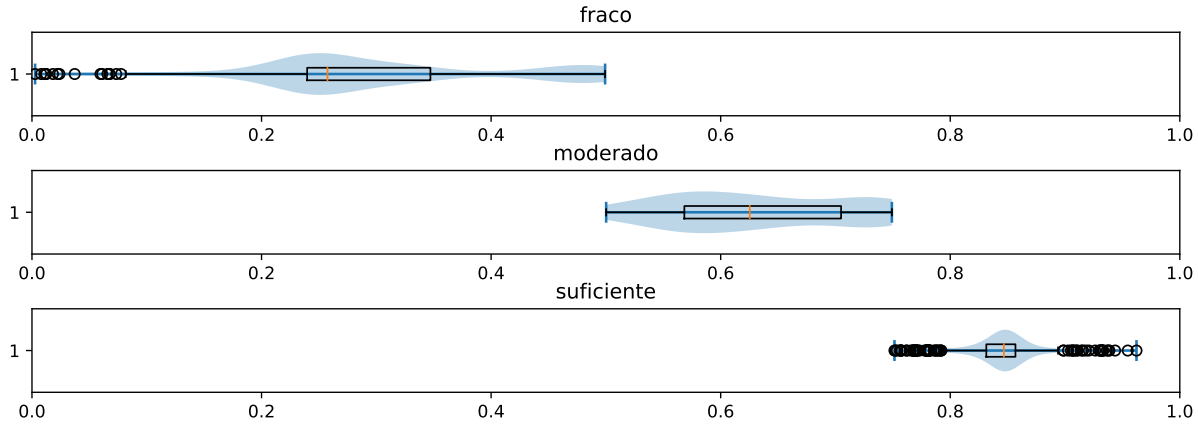


Figura 22 – Distribuição por classificação - Algoritmo FedAVG

Fonte: Produzido pelos autores

em um valor de ± 0.70. Já os modelos "suficientes"se distribuem de forma mais equilibrada em torno do valor 0.85, com uma dispersão menor em relação às outras categorias. Isso indica que esses modelos apresentam um desempenho preditivo mais regular, o que pode-se traduzir em modelos de maior "confiança", atestando assim que o *SFMEI* é capaz de gerar modelos com bom desempenho preditivo através do algoritmo FedAVG.

Entretanto, é necessário avaliar detalhadamente estes modelos quanto ao tipo de modelo e o desempenho por parte dos clientes, não apenas pela classificação, a fim de compreender o desempenho do *SFMEI* quanto a clientes e modelos. Filtrando os modelos que obtiveram os melhores valores de R^2, resultou-se nos registros contidos na Tabela 17. Nela observa-se que os modelos "*model co2*" e "*model water*" foram classificados como "suficientes", através do algoritmo FedAVG. Os demais modelos, foram classificados como "moderados" ou "fracos".

Corroborando com esses dados, na Figura 23 estão ilustrados gráficos de caixa e violino sobrepostos, que demonstram que os modelos "*model co2*" têm escores R^2 que distribuem-se em grande maioria acima de 0.75, classificando-se assim como "suficientes". Já os modelos "*model water*", mesmo tendo várias medidas de escore R^2 classificados como "suficientes", também têm medidas distribuídas abaixo de 0.75, classificando-se assim como "moderados". Da mesma maneira, os modelos "*model energy consumption*" e "*model temperature*" distribuem-se, em grande maioria, com valores classificados como "fracos", ou seja, possuem valor de R^2 abaixo de 0.50.

Tabela 17 – Classificação modelos gerados através do algoritmo de FedAVG.

Modelo	R^2	**Classificação**
client_1_976_model_co2	0.862888	suficiente
client_1_982_model_energy_consumption	0.330649	fraco
client_1_985_model_temperature	0.305570	fraco
client_1_980_model_water	0.938650	suficiente
client_2_995_model_co2	0.882810	suficiente
client_2_99_model_energy_consumption	0.500240	moderado
client_2_994_model_temperature	0.506301	moderado
client_2_989_model_water	0.962275	suficiente
client_3_87_model_co2	0.855590	suficiente
client_3_92_model_energy_consumption	0.343046	fraco
client_3_97_model_temperature	-0.046772	fraco
client_4_996_model_co2	0.878764	suficiente
client_4_9_model_energy_consumption	0.287808	fraco
client_4_998_model_temperature	0.112969	fraco

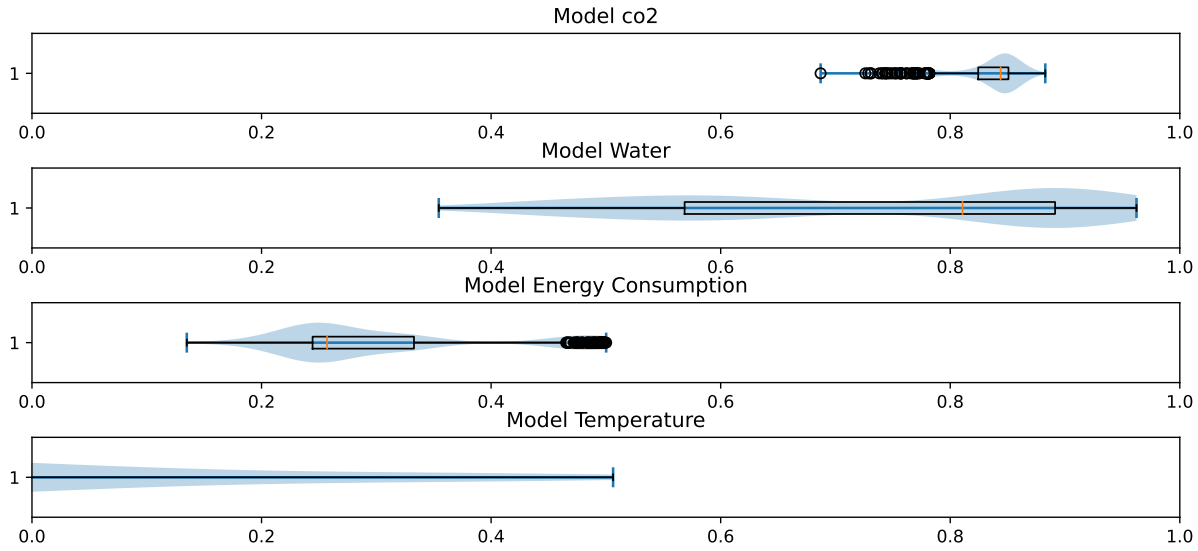


Figura 23 – Distribuição de modelos por classificação - Treinados por todos os clientes - Algoritmo FedAVG

Fonte: Produzido pelos autores

5.2.3.1 FedAVG - Análise MSE por cliente

Entretanto, mesmo que os modelos "*model energy consumption*" e "*model temperature*" tenham sido classificados como "fracos", quanto ao R^2, pode-se, por meio da representação gráfica do comportamento do MSE, analisar como uma métrica complementar para compreender

a convergência desses modelos. Com base na análise dos gráficos presentes nas Figuras 24, 25, 26 e 27, pode-se observar que o cálculo do MSE apresenta uma tendência decrescente ao longo do treinamento, indicando que o modelo manteve-se aprendendo e melhorando sua capacidade de realizar predições ao decorrer das épocas. É possível também observar a relação entre a *Loss* (cálculo do MSE nos dados de treinamento) e a *Evaluation loss* (cálculo do MSE nos dados de avaliação), para avaliar a generalização do modelo, seu desempenho e principalmente evitar sobreajuste, que ocorre quando o modelo se adapta excessivamente apenas aos dados de treinamento, resultando em um desempenho ruim em dados novos e desconhecidos.

Observa-se que a alguns dos modelos gerados aparentam atingir um ponto de convergência, onde a *Loss* e a *Evaluation loss* se estabilizam, indicando que os clientes que geraram estes modelos foram capazes de treinar modelos que convergem, que não possuem viés, nem estão em sobreajuste, mas que, entretanto, possuem um baixo desempenho preditivo. Para analisar estes casos, está em destaque mais escuro na Tabela 17 um recorte dos modelos classificados como "fracos", e "moderados", para o modelo FedAVG. Nela é possível notar que todos os clientes possui uma classificação baixa nos modelos "*model energy consumption*" e "*model temperature*".

Em relação ao cliente 1 (*client 1*), observando a Figura 24 torna-se evidente que o modelo "*model temperature*", que possui valor de $R^2 = 0.30330649$, não demonstra uma estabilização da *Evaluation Loss*, no qual uma discrepância pode observada para estes valores em comparação com a *Loss*. É possível observar que não houve redução significativa do valor de MSE durante todas as épocas de treinamento, o que demonstra que este modelo não foi capaz de manter-se aprendendo no decorrer das épocas neste cliente. Já em relação ao modelo "*model energy consumption*" é notório um comportamento de bastante inconsistência da *Loss* e *Evaluation loss*, demonstrando um não equiblíbrio no decorrer das épocas, o que também é possível observar no modelo "*model co2*", mesmo que o mesmo para este cliente, seja classificado como "suficiente".

Em relação ao cliente 2 (*client 2*), observando a Figura 25 percebe-se que embora com desempenho baixo, o modelo "*model energy consumption*" demonstram além de uma aparente estabilização, não demonstra também um sobreajuste e percebe-se uma homogeneidade da *Loss* e da *Evaluation Loss*. Também para este cliente, torna-se evidente o mesmo comportamento do cliente 1, para o modelo "*model temperature*".

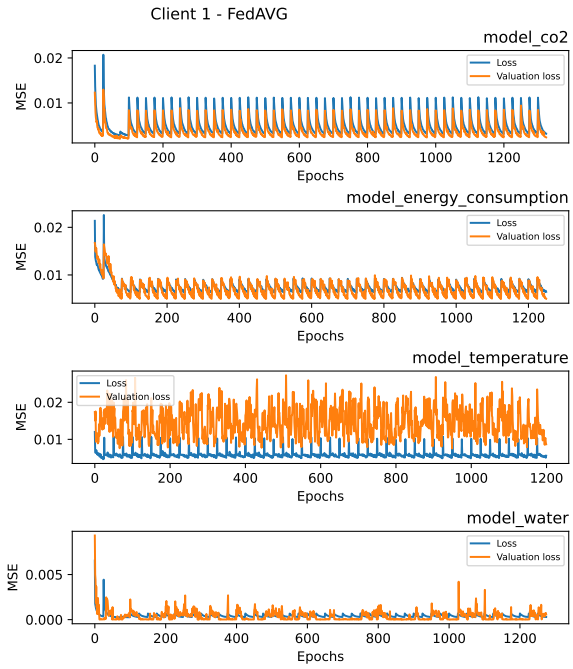


Figura 24 – MSE - Cliente 1 - Todos os modelos - Algoritmo FedAVG

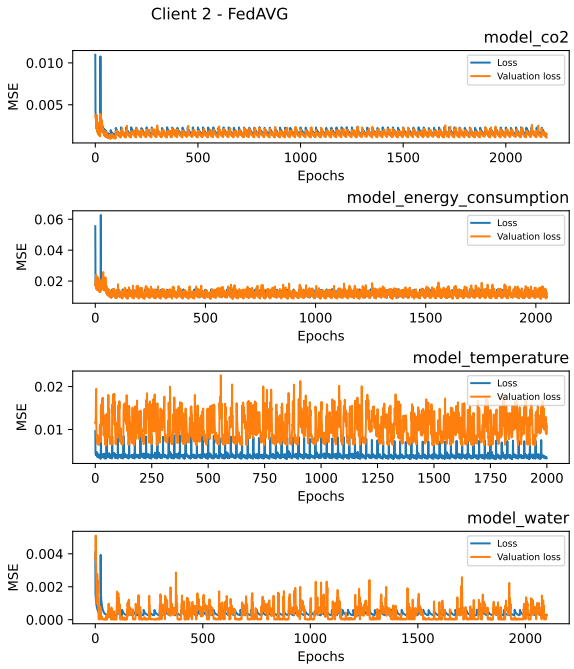


Figura 25 – MSE - Cliente 2 - Todos os modelos - Algoritmo FedAVG

Fonte: Produzido pelos autores

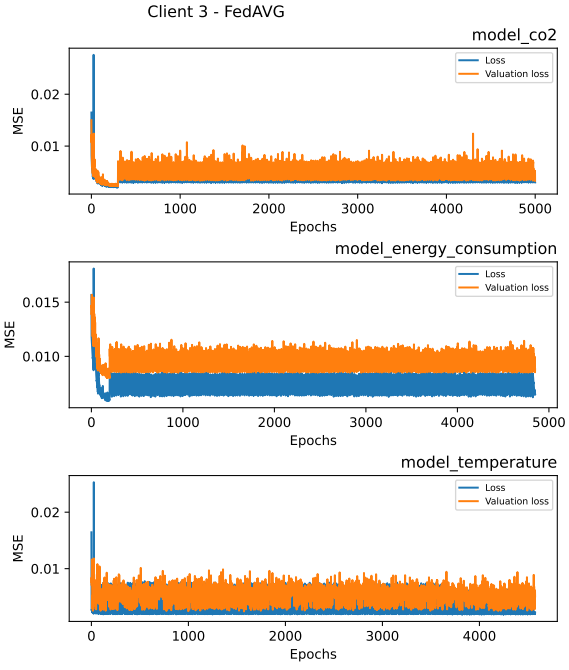


Figura 26 – MSE - Cliente 3 - Todos os modelos - Algoritmo FedAVG

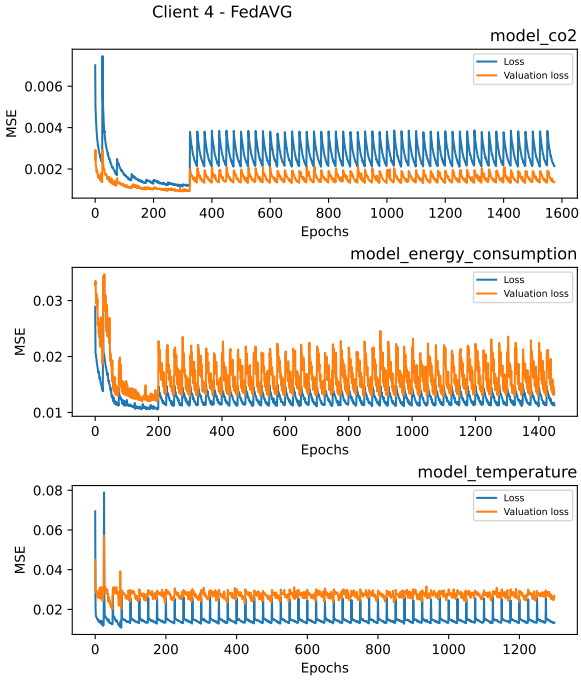


Figura 27 – MSE - Cliente 4 - Todos os modelos - Algoritmo FedAVG

Fonte: Produzido pelos autores

De maneira similar, entre os clientes 3 (*client 3*), Figura 26, e 4 (*client 4*), Figura 27, não é possível identificar de maneira clara que a *Loss* e a *Evaluation Loss* se estabilizam, com

tendência a zero, indicando que os clientes que geraram estes modelos foram capazes de treinar modelos que convergem, que não possuem viés, nem estão em sobreajuste.

Desta maneira não é possível afirmar com clareza que todo cliente participante do *SFMEI* é capaz de obter bons desempenhos e bons resultados durante a geração de modelos ao utilizar o algoritmo FedAVG. Este aspecto é essencial para compreender os possíveis prejuízos de um cliente participar *SFMEI*, sem que haja uma possibilidade de contribuir com um bom potencial preditivo.

5.2.3.2 FedAVG - Análise MSE por modelo

Os resultados expostos são incertos quanto a demonstrar que todos os clientes, para o algoritmo FedAVG, são capazes de gerar modelos que possuem bom desempenho. Entretanto, podemos realizar uma análise auxiliar quanto ao desempenho de cada modelo. Por meio destes resultados é, possível observar que os modelos "model co2" e "model water" são os únicos que não possuem uma classificação baixa para o algoritmo FedAVG.

Por meio dos resultados observados até aqui, pode-se verificar que o modelo "model co2", apenas para o cliente 2, demonstra haver uma estabilização nos valores do MSE, além de que é evidente não haver um sobreajuste, não identificando um comportamento de viés. Embora este comportamento não seja observado para os demais clientes, o modelo "model co2", de acordo com os valores calculados de R^2, foram classificados como suficientes, mesmo que para este caso haja um comportamento incomum quanto aos cálculos da função de perda.

Já em relação ao modelo "*model energy consumption*", que possui uma baixa classificação em todos os clientes observa-se que os valores do cálculo do MSE para este modelo não possuem uma tendência a zero, o que pode demonstrar que o mesmo não foi capaz de manter-se aprendendo ao longo das épocas de treinamento. O mesmo pode-se observar quanto "*model temperature*", em que os valores da *Loss* e *Evaluation Loss* em nenhum dos clientes demonstra uma tendência a zero. Por estes fatores abre-se também nestes casos uma oportunidade de análise futura a fim de melhor avaliar e compreender o desempenho do *SFMEI* e seus clientes no treinamento destes modelos por meio do algoritmo FedAVG.

5.2.3.3 FedAVG - Modelos agregados finais

No algoritmo FedAVG, como detalhado anteriormente, são gerados modelos agregados a cada rodada de aprendizagem no *SFMEI*. Estes modelos agregados são calculados a partir da média dos modelos recebidos pelos clientes a cada rodada de aprendizagem federada. Os modelos agregados melhor classificados são disponibilizados no repositório de modelos do *SFMEI*.

Na Tabela 18 estão descritos os modelos agregados melhores classificados, divididos por conjunto de dados. Pelo *SFMEI* estes seriam os modelos disponibilizados em seu repositório. É

possível identificar que os modelos agregados pelo servidor refletem o mesmo comportamento dos modelos analisados anteriormente, gerados pelos clientes. Os modelos "*model co2*" e "*model water*" destacam-se classificados como "suficientes".

Tabela 18 – Classificação modelos finais agregados através do algoritmo de FedAVG.

Modelo	R^2	**Classificação**
server_98_model_co2	0.773708	suficiente
server_99_model_energy_consumption	0.341075	fraco
server_9_model_temperature	0.493405	fraco
server_87_model_water	0.865431	suficiente

Estes aspectos evidenciam que o *SFMEI*, aplicando o algoritmo FedAVG, teve um bom desempenho em alguns casos e outros não. No caso do modelo "*model energy consumption*", que foi classificado como "fraco", mas que demonstra possível convergência, abre-se uma oportunidade de realizar pré-processamentos, tratamentos e aplicar algoritmos que foram deixados de fora deste trabalho, como citados na Seção 3.3.1. Para o modelo "*model temperature*" não há evidência direta de que pode haver outros aspectos e ajustes de parâmetros a seres aplicados para buscar melhores métricas de execução.

5.2.4 Resultados cenário de teste 2 - FedSGD

Na Figura 28 é mostrada a quantidade de modelos gerados por classificação no algoritmo FedSGD. Foi possível notar que a grande maioria dos modelos gerados dividem-se entre "suficientes" e "fracos", assim como no FedAVG. Por meio da Tabela 19 pode-se observar que a execução do *SFMEI* aplicando um algoritmo de FedSGD obteve resultados melhores que a execução com o algoritmo FedAVG. No FedSGD, 47,12% foram classificados como "suficientes", e os modelos 'fracos" representam 46,90%. Por fim, os modelos "suficientes" no FedSGD possuem uma mediana de escore $R^2 = 0,86$.

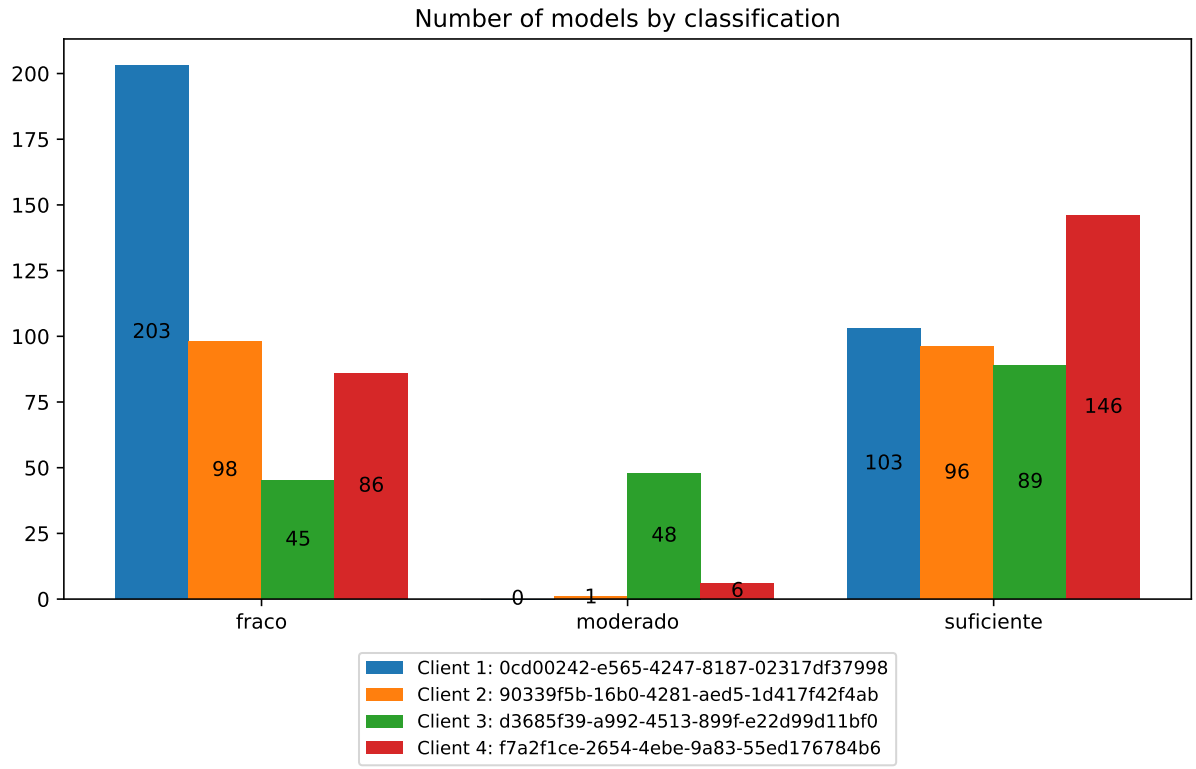


Figura 28 – Número de modelos por classificação - Treinados por todos os clientes - Algoritmo FedSGD

Fonte: Produzido pelos autores

Tabela 19 – Distribuição de frequências de modelos gerados - FedSGD

Classificação	**Quantidade**	**Percentual Classificação**	**Mediana R^2**
Fraco	432	46,90%	0.374335
Moderado	55	05,97%	0.632094
Suficiente	434	47,12%	0.863973

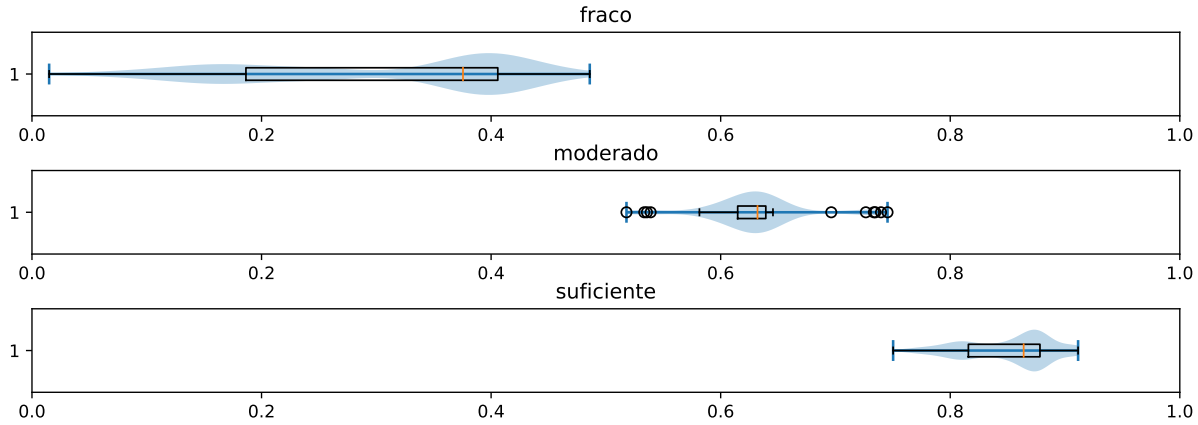


Figura 29 – Distribuição por classificação - Algoritmo FedSGD

Fonte: Produzido pelos autores

Analisando a distribuição dos escores R^2 por classificação, podemos encontrar na Figura 29 uma ilustração que auxilia na análise outros aspectos destes modelos. Os modelos classificados como "fracos", acima, uma concentração abaixo, mas aproximados, do limiar para a classificação "moderado", pois segundo quartil (mediana) e terceiro quartil estão posicionada em um valor de ± 0.4. Os modelos "moderados", ao centro, se distribuem de forma mais equilibrada entre 0.60 e 0.65, embora representem $\pm 5\%$ do total de modelos gerados por esse algoritmo. Entretanto é possível também observar os modelos desta classificação se mantém ainda distante do limiar de “suficiente”. Os modelos "suficientes", abaixo, se distribuem de forma mais variada entre ± 0.82 ± 0.88, sendo bem dispersos. Isso indica que esses modelos apresentam um bom desempenho preditivo mesmo que não tão regular, mas que, de toda maneira atestar a capacidade do *SFMEI* de gerar modelos com bom desempenho preditivo por meio do algoritmo FedSGD.

Filtrando também pelos modelos que obtiveram o melhor valor de R^2, resultou-se nos registros contidos na Tabela 20. Nela observa-se que os modelos “*model co2*” e “*model temperature*” foram classificados como “suficientes”, por meio deste algoritmo, FedSGD, enquanto os demais foram classificados como “moderados” ou “fracos”.

A Figura 30 ilustra esses resultados. Através dos gráficos de caixa e violino sobrepostos, é possível demonstrar que os modelos “*model co2*” têm escores R^2 que distribuem-se acima de 0.75, classificando-se como “suficientes”. Já os modelos “*model temperature*”, mesmo tendo várias medidas de escore R^2 classificados como “suficientes”, também têm medidas distribuídas abaixo de 0.75, classificando-se assim como “moderados”. As medidas dos modelos “*model energy consumption*” distribuem-se, em grande maioria, com valores classificados como “fracos”, salvo os casos dos limites superiores do gráfico de caixa, que representam casos melhor avaliados, que classificaram-se como “moderados”, ou seja, um valor de R^2 acima de 0.50. Por fim, os modelos “*model water*” possuem todos os valores de R^2 considerados como “fracos”, abaixo de 0.50.

Tabela 20 – Classificação modelos gerados através do algoritmo de FedSGD.

Modelo	R^2	**Classificação**
client_1_90_model_co2	0.814739	suficiente
client_1_830_model_energy_consumption	0.200633	fraco
client_1_837_model_temperature	0.882986	suficiente
client_1_832_model_water	0.378494	fraco
client_2_914_model_co2	0.911401	suficiente
client_2_94_model_energy_consumption	0.645562	moderado
client_2_99_model_temperature	0.897768	suficiente
client_2_97_model_water	0.254660	fraco
client_3_91_model_co2	0.872753	suficiente
client_3_95_model_energy_consumption	0.198714	fraco
client_3_93_model_temperature	0.904275	suficiente
client_4_96_model_co2	0.879831	suficiente
client_4_9_model_energy_consumption	0.421223	fraco
client_4_920_model_temperature	0.485951	fraco

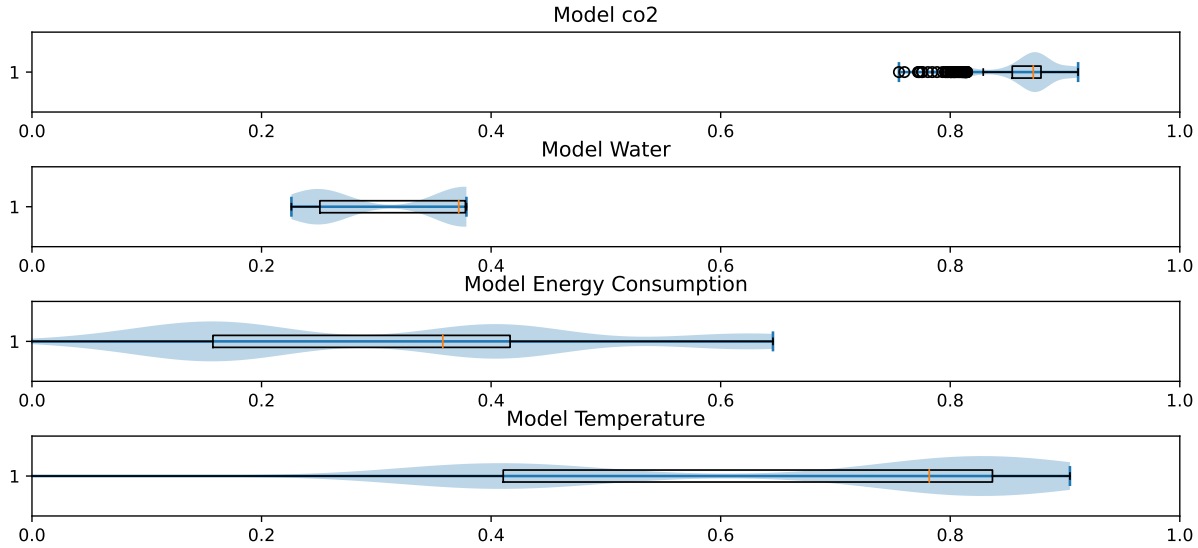


Figura 30 – Distribuição de modelos por classificação - Treinados por todos os clientes - Algoritmo FedSGD

Fonte: Produzido pelos autores

5.2.4.1 FedSGD - Análise MSE por cliente

Entretanto, mesmo que os modelos "*model energy consumption*" e "*model water*" tenham sido classificados como "fracos", quanto ao R^2, pode-se, através da representação gráfica do comportamento do MSE, analisar como uma métrica complementar para compreender a convergência desses modelos. Com base na análise dos gráficos presentes nas Figuras 31, 32, 33 e 34, pode-se

observar que o MSE apresenta uma tendência decrescente ao longo do treinamento, indicando que o modelo manteve-se aprendendo e melhorando sua capacidade de realizar predições ao decorrer das épocas. É possível também observar a relação entre a *Loss* (cálculo do MSE nos dados de treinamento) e a *Evaluation Loss* (cálculo do MSE nos dados de avaliação), para avaliar a generalização do modelo, seu desempenho e principalmente evitar sobreajuste, que ocorre quando o modelo se adapta excessivamente apenas aos dados de treinamento, resultando em um desempenho ruim em dados novos e desconhecidos.

Assim, observa-se que a maioria dos modelos gerados por todos os clientes aparentam atingir um ponto de convergência, onde a *Loss* e a *Evaluation Loss* se estabilizam , o que indica que os clientes foram capazes de treinar modelos que convergem, não possuem viés, nem estão em sobreajuste, mas que em alguns casos possuem um baixo desempenho preditivo. Os dados em destaque na Tabela 20 demonstram um recorte dos modelos classificados como "fracos", e "moderados", para que seja possível analisá-los quanto ao MSE.

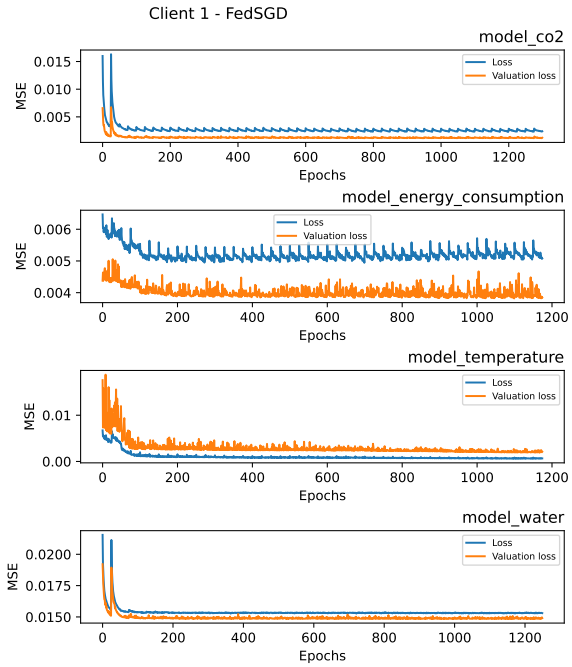


Figura 31 – MSE - Cliente 1 - Todos os modelos - Algoritmo FedSGD

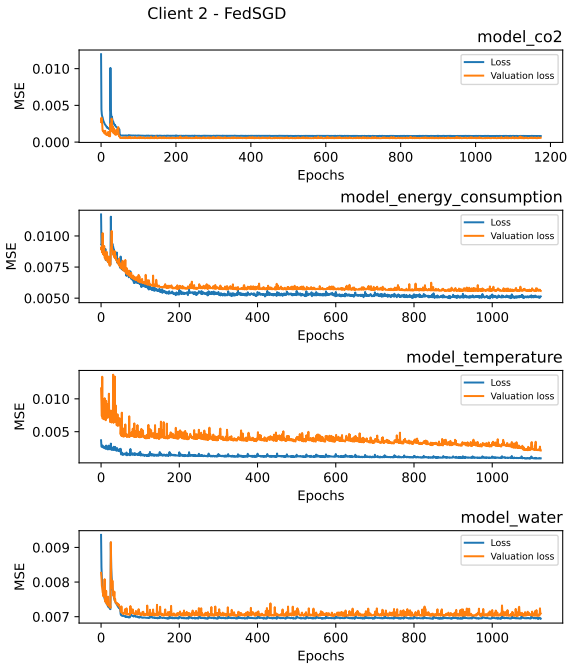


Figura 32 – MSE - Cliente 2 - Todos os modelos - Algoritmo FedSGD

Fonte: Produzido pelos autores

O cliente 1 (*client 1*) possui uma classificação baixa nos modelos "*model energy consumption*" e "*model water*". Observando na Figura 31 torna-se evidente que o modelo "*model energy consumption*", que possui valor de $R^2 = 0,200633$, não aparenta um sobreajuste, mas que, entretanto, demonstra uma estabilização da *Loss* e da *Evaluation Loss*, uma discrepância expressiva é observada nestes valores em escala. É possível observar que não houve redução significativa do valor de MSE durante todas as épocas de treinamento, o que demonstra que este modelo não foi capaz de manter-se aprendendo no decorrer das épocas neste cliente. Já em

relação ao modelo "*model water*" não é demonstrado um aparente mal comportamento da *Loss* e *Evaluation Loss*, entretanto, mais detalhe serão abordados na seção 5.2.5.

O cliente 2 (*client 2*) possui uma classificação baixa também nos modelos "*model energy consumption*" e "*model water*". Entretanto, observando a Figura 32 percebe-se que embora com desempenho baixo, ambos os modelos demonstram além de uma estabilização, também a não existência de sobreajuste e uma homogeneidade da *Loss* e da *Evaluation Loss*, que demonstrado através da redução ao longo das épocas, manteve-se aprendendo. Demonstrando também para estes modelos que podem existir outros aspectos que contribuíram para o baixo desempenho, mas que os mesmos possuem um bom potencial de melhoria.

Estes mesmos aspectos dos modelos do cliente 2 aplicam-se para o cliente 3 (*client 3*), Figura 33, que possui um baixo desempenho no modelo "*model energy consumption*", com $R^2 = 0,198714$, e para o cliente 4 (*client 4*), Figura 34, que além de também possuir um baixo desempenho no modelo "*model energy consumption*", com $R^2 = 0,198714$, o modelo "*model temperature*", com $R^2 = 0,455951$, também é classificado como "fraco". Uma vez que estes modelos possuem as mesmas características similares aos observado no cliente 2, o potencial de melhoria possível a eles também se aplica neste caso.

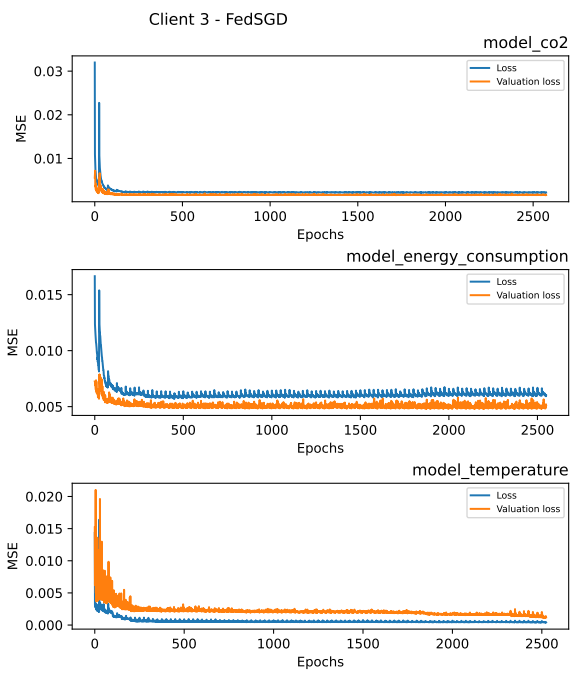


Figura 33 – MSE - Cliente 3 - Todos os modelos - Algoritmo FedSGD

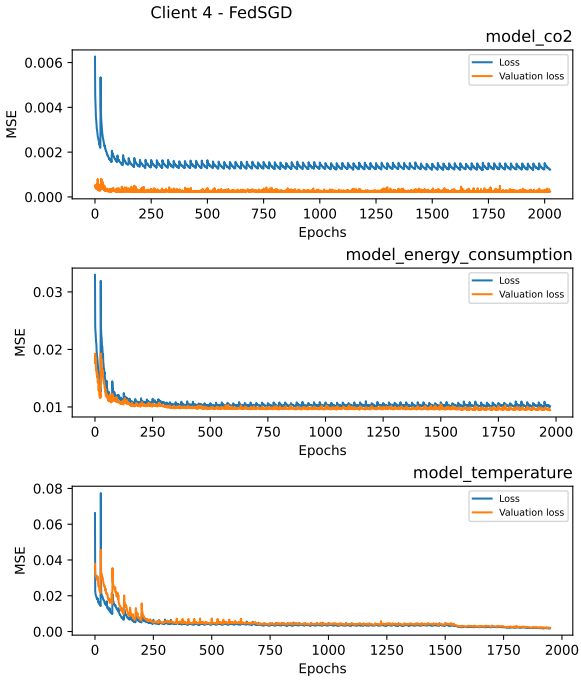


Figura 34 – MSE - Cliente 4 - Todos os modelos - Algoritmo FedSGD

Fonte: Produzido pelos autores

Desta maneira é possível afirmar que todo cliente participante do *SFMEI* é capaz de obter bons desempenhos e bons resultados durante a geração de modelos ao utilizar o algoritmo FedSGD. Este aspecto é essencial para que não haja prejuízos ao *SFMEI* as possuir distintos

clientes participantes. O potencial de melhoria possível aos modelos que possuíram desempenhos baixos também demonstra-se como um ponto positivo ao *SFMEI*, uma vez que isso pode, por exemplo, auxiliar em uma avaliação da qualidade da contribuição por cliente, em uma evolução do mesmo.

5.2.4.2 FedSGD - Análise MSE por modelo

Os resultados expostos até aqui demonstram que todos os clientes, para o algoritmo FedSGD, são capazes de gerar modelos que possuem bom desempenho, entretanto, podemos também realizar uma análise quanto ao desempenho de cada modelo. Por meio destes dados é possível observar que o modelo "*model co2*" é o único a não possuir uma classificação baixa. Já o modelo "*model energy consumption*" é o único modelo a possuir baixa classificação em todos os clientes. Complementam estes aspectos, por exemplo, os resultados dos modelos "*model co2*", em que, para todos os clientes, demonstram haver uma estabilização nos valores do MSE, além disso, também não demonstrando a existência de um viés ou sobreajuste. Também observa-se uma homogeneidade da *Loss* e da *Evaluation Loss*, que, reduzindo ao longo das épocas até atingirem um ponto de convergência e equilíbrio.

Observa-se também, através do comportamento dos valores do MSE para o modelo "*model energy consumption*", que possui uma baixa classificação em todos os clientes, que os valores da *Loss* e *Evaluation Loss* deste modelo não possuem uma tendência a zero. Além disso, estes modelos se mantém em equilíbrio há muitas épocas, sem muitas variações, não demonstrando um viés, mas que pode pode ser observado como sendo inalterável o potencial de predição deste modelo, ou seja, ainda baixo. Desta forma abre-se a oportunidade de melhoria e outras análises futuras para compreender a razão do baixo desempenho do *SFMEI* e seus clientes no treinamento deste modelo.

Observa-se com modelo "*model temperature*" um comportamento singular dos demais analisados. Neste modelo, é possível enxergar que mesmo após muitas épocas o modelo treinado ainda possui uma tendência decrescente e mantém-se em aparente redução, ou seja, este modelo ainda seria capaz de ter um melhor resultado que o observado. O prolongamento deste aprendizado, por mais épocas, com um ponto de interrupção prolongado, poderia resultar em um modelo com um melhor potencial preditivo do que o apresentado.

5.2.4.3 FedSGD - Modelos agregados finais

No algoritmo FedSGD, conforme a definição abordada anteriormente, são definidos os modelos agregados a cada rodada de aprendizagem no *SFMEI*. Estes modelos agregados são definidos a partir da avaliação dos melhores modelos gerados pelos clientes a cada rodada de aprendizagem federada. Cada melhor modelo é enviado para rodada de aprendizagem. Os modelos melhor classificados são disponibilizados no repositório do *SFMEI*.

Na Tabela 21 estão detalhados os modelos melhores classificados, divididos por conjunto de dados. Pelo *SFMEI*, estes seriam os modelos disponibilizados em seu repositório. Estes modelos finais refletem o comportamento dos modelos analisados anteriormente. É possível observar que os modelos "*model temperature*" e "*model co2*" destacam-se classificados como "suficientes". Além deles, o modelo "*model energy consumption*" também possui um modelo final classificado como "suficiente", conforme o evidenciado pela Figura 30.

Tabela 21 – Classificação modelos finais agregados através do algoritmo de FedSGD.

Modelo	R^2	**Classificação**
client_0_832_model_water	0.378494	fraco
client_1_93_model_temperature	0.904275	suficiente
client_2_914_model_co2	0.911401	suficiente
client_2_94_model_energy_consumption	0.645562	moderado

Estes aspectos evidenciam que, aplicando o algoritmo FedSGD no *SFMEI*, há o potencial de que se obtenha um bom desempenho em praticamente todos os casos. No caso dos modelo "*model energy consumption*" e "*model water*", mesmo classificados como "suficiente" e "fraco", respectivamente, o MSE demonstra uma possível convergência, o que abre a oportunidade e possibilidade de realizar pré-processamentos, tratamentos e algoritmos adicionais de ajuste que foram deixados de fora deste trabalho, como citados na Seção 3.3.1, com a finalidade de aprimorar o desempenho final desses modelos.

5.2.5 Resultados cenário de teste 2 - Número reduzido de clientes

Com a finalidade de obter outros resultados para análise do desempenho geral do *SFMEI* optou-se, de forma arbitrária, por escolher um dos modelos a ser treinado por um número reduzido de clientes, a fim de também compreender e considerar o impacto do número de clientes na qualidade dos modelos gerados pelo *SFMEI*. Este modelo foi o "*model water*", definido devido a esta variável possuir o maior volume de dados a ser utilizado para treinamento. Esta escolha se deu para que fosse possível ir além e, de forma complementar, avaliar a capacidade do *SFMEI* durante a geração de modelos utilizando um número reduzido de nós participantes durante o treinamento.

O número reduzido de clientes foi definido de modo que o modelo fosse exposto a apenas um conjunto de dados distinto. Desta forma, o modelo "*model water*" foi treinado por apenas dois clientes, cliente 1 e 2 (*client 1* e *client 2*). Assim, os modelos treinados localmente, após agregados, seriam expostos a apenas um conjunto de dados distinto. Desta maneira buscou-se identificar se este modelo conservaria bons resultados.

Na execução do algoritmo FedAVG foi possível observar que, os maiores valores de R^2 desse modelo foram classificados como "suficientes", enquanto no algoritmo FedSGD, os maiores valores desse modelo foram classificados como "fracos", para ambos os clientes que

realizaram o treinamento destes modelos. Para o algoritmo FedAVG, foi possível obter modelos com valor de $R^2 = 0.938659$, para o primeiro cliente e $R^2 = 0.962275$, para o segundo cliente. Enquanto que para o algoritmo FedSGD, mesmo os maiores valores obtidos de R^2, foram de apenas 0.0.378494 e 0.254660, como detalhado na Tabela 22.

Tabela 22 – Classificação modelos “model water”.

Modelo	R^2	**Classificação**	**Algoritmo**
client_0_832_model_water	0.378494	fraco	FedSGD
client_2_97_model_water	0.254660	fraco	FedSGD
client_0_980_model_water	0.938650	suficiente	FedAVG
client_3_989_model_water	0.962275	suficiente	FedAVG

Já em relação ao modelo “model water”, Figura 35, gerados pelo algoritmo FedSGD é possível observar que, mesmo com a baixa avaliação de $R^2 = 0.378494$, este modelo demonstra uma estabilização a não existência de sobreajuste e também a existência uma homogeneidade da *Loss* e da *Evaluation Loss*, juntamente com uma redução expressiva inicial que demonstra que o modelo manteve-se aprendendo inicialmente, mas em seguida estabilizou-se sem possuir bom desempenho. Isso demonstra que podem existir outros aspectos que contribuíram para o baixo desempenho deste modelo e que o mesmo possui um bom potencial de melhoria.

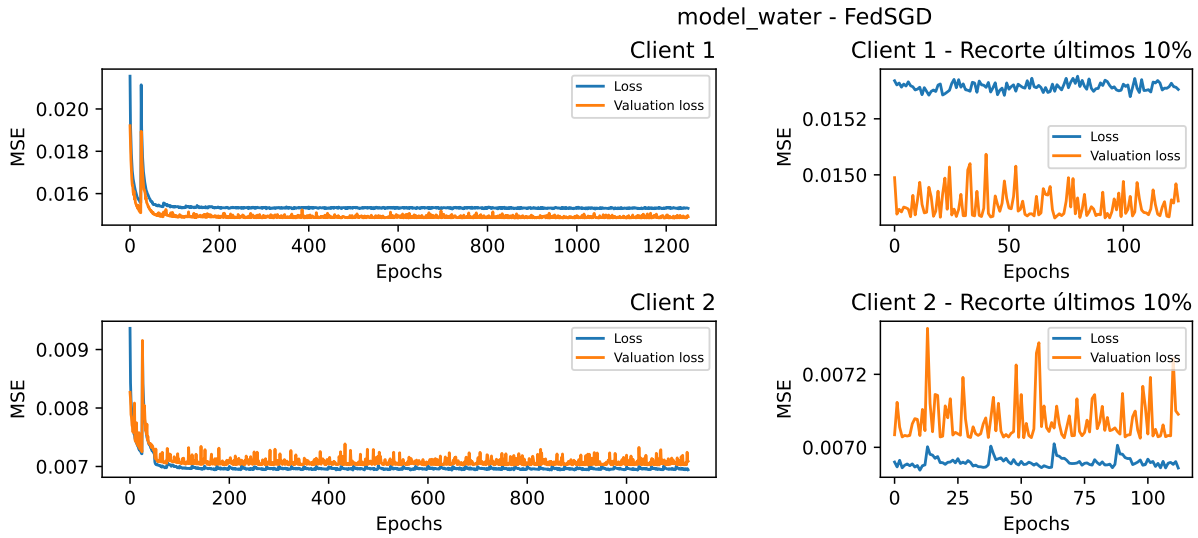


Figura 35 – MSE - Modelo “model water” - Algoritmo FedSGD

Fonte: Produzido pelos autores

5.2.6 Resultados cenário de teste 2 - Comparativo MSE

Através de todos os resultados do cenário de teste 2 expostos até o momento, é possível verificar que um algoritmo se sobressai sobre o outro. O algoritmo FedSGD para os modelos “*model co2*”, “*model energy consumption*” e “*model temperature*” aparenta possuir melhores

resultados, quando comparação ao algoritmo FedAVG. Já o FedAVG aparenta ter melhor resultado para o modelo "*model water*". A Tabela 23 ilustra o cálculo da mediana para todos os modelos gerados em ambos os algoritmos, através dela é possível comprovar esta observação. Desta forma é possível afirmar que o *SFMEI* possui um melhor desempenho quando utilizado o algoritmo FedSGD, mesmo que sejam gerados menos modelos, no mesmo período de tempo.

Tabela 23 – Mediana do cálculo do MSE por modelo para os algoritmos FedAVG e FedSGD.

Modelo	**Mediana MSE FedAVG**	**Mediana MSE FedSGD**
client_1_model_co2	0.004219	0.002545
client_1_model_water	0.000363	0.015325
client_1_model_energy_consumption	0.007295	0.005151
client_1_model_temperature	0.005627	0.000826
client_2_model_co2	0.001699	0.000842
client_2_model_water	0.000328	0.006962
client_2_model_energy_consumption	0.010842	0.005267
client_2_model_temperature	0.003691	0.001217
client_3_model_co2	0.003471	0.002234
client_3_model_energy_consumption	0.007102	0.006024
client_3_model_temperature	0.002173	0.000498
client_4_model_co2	0.002475	0.001360
client_4_model_energy_consumption	0.012012	0.010301
client_4_model_temperature	0.014177	0.003631

5.2.7 Resultados do cenário de teste 3 - Avaliação do desempenho do *SFMEI* provendo modelos pré-treinados

De acordo com o afirmado na Seção 5.2.6, em que o *SFMEI* possui melhor desempenho quando utilizado o algoritmo FedSGD, um cenário de simulação foi criado a fim de emular um caso em que clientes conectados ao *SFMEI* não sejam participantes, mas sim consumidores dos modelos pré-treinados. Três clientes hipotéticos (Clientes X, Y e Z) foram simulados como não participantes de treinamento, mas seriam capazes de consumir modelos preditivos. Estes clientes buscaram no repositório de modelos pré-treinados do *SFMEI* o modelo de melhor desempenho, no caso o "*model co2*" ($R^2 = 0,911401$), obtido a partir do algoritmo FedSGD. O *script* utilizado para simular estes clientes está presente e detalhado no Apêndice F, que simula um cenário em que cada cliente utiliza localmente a interface pública, do *SFMEI Client*, para buscar o modelo via chamadas HTTP, em seguida carrega este modelo localmente e utiliza dados locais para realizar predições.

Cada um dos clientes utilizou amostras distintas de medidas de CO_2, também parte do conjunto de dados *Smart Campus Oulu indoor climate, air-quality and motion* (Oulu, 2023) para avaliação. Estas amostras de dados diferenciam-se tanto umas das outras, como também

das utilizadas durante a aprendizagem federada. Essas amostras foram assumidas com dados de teste a serem utilizados como referência para o modelo preditivo, as quais, ao final poderiam ser comparadas de maneira direta com os dados previstos pelo modelo. Após cada predição realizada por cada cliente foi avaliado o valor de R^2 obtido, juntamente com as métricas MSE, MAE e RMSE, de modo a avaliar o desempenho preditivo e adaptabilidade deste modelo em casos em que o mesmo não tenha sido exposto em nenhum momento aos dados a serem previstos.

Na Tabela 24 é possível observar os resultados detalhados obtidos por esses clientes, bem como uma classificação destes resultados. Através deles é possível verificar que o modelo, para os clientes X e Z obtive um valor de R^2 classificado como "suficientes", já para o cliente Y foi classificado como "moderado". As demais métricas do MAE, MSE e RMSE possuem, por definição um melhor desempenho quanto mais aproximado de 0. Com isso, é possível observar que o modelo, para o cliente Y possui cálculos de erro com melhor resultado do que para o cliente X, demonstrando que, mesmo não possuindo uma classificação "suficiente", seu potencial preditivo para este cliente pode ser equiparado aos demais.

Tabela 24 – Classificação modelo "*model co2*" consumido pelos clientes X, Y e Z.

Cliente	**R^2 score**	**Classificação**	**MAE**	**MSE**	**RMSE**
X	0.762735	suficiente	0.030512	0.001555	0.039437
Y	0.644880	moderado	0.022404	0.000884	0.029738
Z	0.763156	suficiente	0.017261	0.000627	0.025038

Para uma análise mais detalhada, as Figuras 36, 37 e 38 ilustram o comparativo entre os dados de teste locais e os dados resultantes da predição utilizando o modelo pré-treinado "*model co2*". Nelas é possível observar análises gráficas que visam demonstrar que o comportamento preditivo se assemelhe aos dados de teste aplicados. Nos gráficos à esquerda e ao centro, é possível observar em azul o registro da medida dos dados de teste utilizados por cada um dos clientes e em laranja os dados gerados a partir da predição. É possível perceber, nos gráficos a esquerda, que em todas elas o comportamento preditivo se assemelha e reproduz com bastante similaridade os dados de teste utilizados como referência. Já observando os gráficos ao centro é possível verificar a distribuição dos dados de teste, em comparação aos dados previstos, neles é possível observar mais uma vez a semelhança e baixa discrepância entre ambas as distribuições, de modo que, em alguns casos, aparentam até sobrepor-se umas as outras. À direita estão traçadas as retas de regressão entre os valores calculados e os valores previstos. Por meio deles é possível verificar o motivo que reduziu o valor do cálculo do R^2 para o cliente Y, onde, na Figura 37 observa-se que a reta traçada possui uma inclinação menor, em diagonal, quando comparada as retas dos outros clientes. Quanto menos inclinada for a reta de regressão, mais aproximando-se de uma reta paralela ao eixo X, maior é o erro observável entre os dados, entretanto, mesmo não possuindo uma inclinação diagonal como as demais, o cliente Y possui pontos de similaridade e uma baixa discrepância dos dados, aproximando-se muito, em grande parte dos casos, a reta que

descreve estes dados. Desta maneira, mesmo possuindo alguns erros que se distanciam bastante da reta, neste cliente o modelo demonstra um bom potencial preditivo para uma boa parcela dos casos. Desta forma é possível inferir que um modelo pré-treinado através do *SFMEI* de maneira federada é capaz de possuir um desempenho preditivo bom em distintos clientes, com distintos dados a serem previstos.

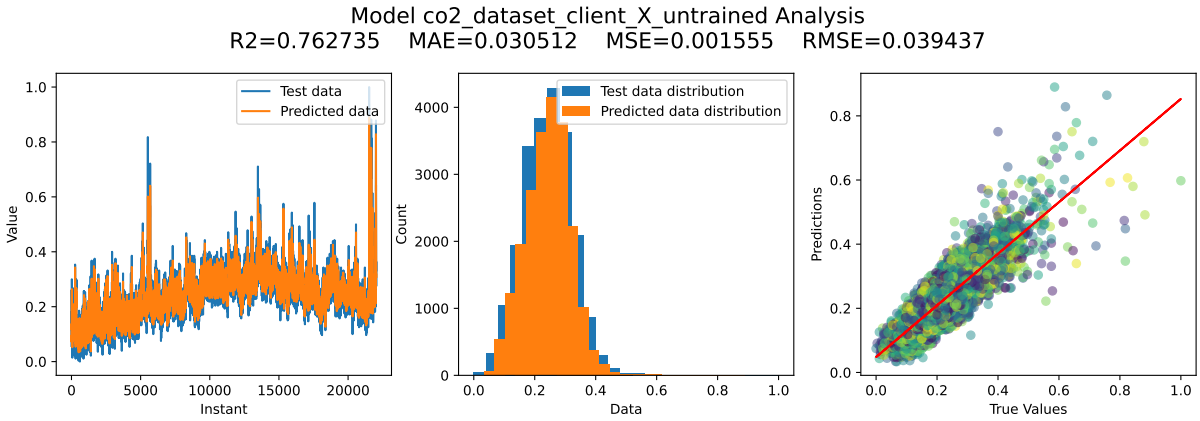


Figura 36 – Predição "*model co2*" cliente X

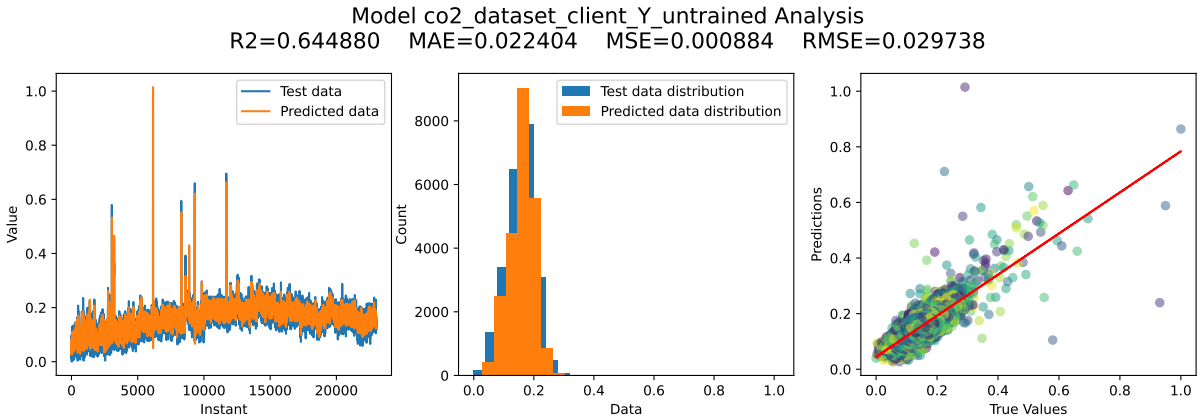


Figura 37 – Predição "*model co2*" cliente Y

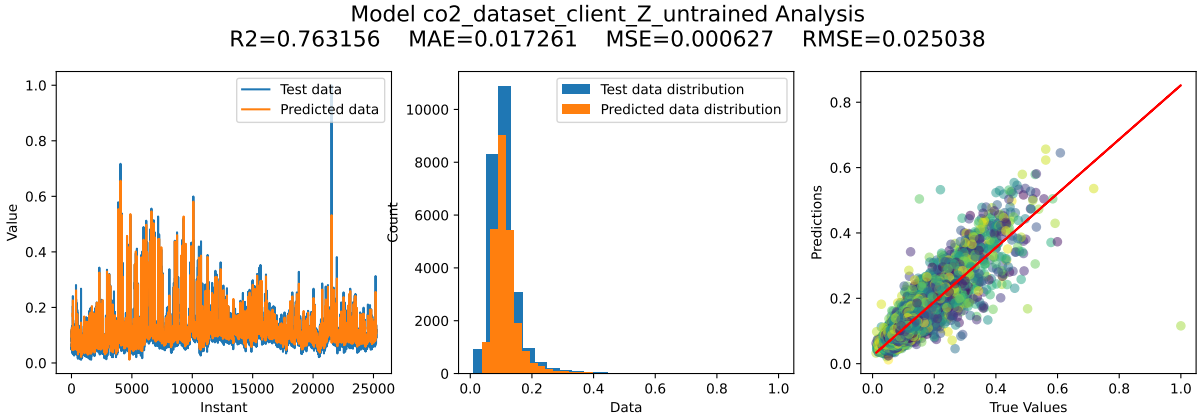


Figura 38 – Predição "*model co2*" cliente Z

Fonte: Produzido pelos autores

5.3 Discussão

5.3.1 Quanto à execução dos cenários de teste

Com base nos resultados da primeiro cenário de teste, é possível responder a questão de pesquisa **"QP1) Há prejuízos em utilizar o *middleware* proposto em comparação com uma abordagem de aprendizagem de máquina não federada?"**, pois não há prejuízos em utilizar o SFMEI em comparação com uma abordagem de aprendizagem de máquina não federada. O protótipo do SFMEI, utilizando o algoritmo de agregação FedAVG, foi capaz de alcançar um valor de escore $R^2 = 0,9898$ para a predição do consumo de energia, similar ao valor de referência de $R^2 = 0,9895$ obtido pela aplicação não federada. Estes resultados demonstram que o SFMEI é capaz de preservar o potencial preditivo de modelos de aprendizado de máquina, mesmo em um ambiente federado onde os dados são distribuídos e a privacidade dos nós é preservada. Essa equivalência no desempenho indica que o SFMEI pode ser utilizado como uma alternativa viável para aplicações que exigem privacidade e segurança de dados, sem comprometer a precisão dos modelos.

Além disso, o SFMEI oferece maior escalabilidade e flexibilidade, permitindo o treinamento de modelos em larga escala e a adaptação a diferentes tipos de dados e aplicações. Em resumo, o SFMEI se apresenta como uma alternativa promissora para o desenvolvimento de soluções de aprendizado de máquina em ambientes que exigem privacidade e segurança de dados, sem comprometer a precisão dos modelos.

Quanto aos demais cenários de teste foi possível, por meio de uma melhor análise dos resultados da execução, observar que para o algoritmo FedAVG, em dois conjuntos de dados e dois modelos gerados, o desempenho do *SFMEI* demonstrou-se bom. Embora o mesmo seja observado na execução do cenário de teste do algoritmo FedSGD, no qual para apenas dois conjuntos de dados o desempenho demonstrou-se bom, outros aspectos evidenciam um melhor desempenho nesse algoritmo. Utilizando o FedSGD é possível que sejam levantados pontos que possam futuramente influenciar positivamente nos resultados para os modelos classificados como "fracos".

Conforme detalhado na Tabela 25, embora o algoritmo FedAVG tenha demonstrado resultados "fracos" para os modelos "*model energy consumption*" e "*model temperature*", foi possível obter resultados "suficientes" para os modelos "*model co2*" e "*model water*". De forma semelhante ocorreu para o algoritmo FedSGD, onde os modelos "*model temperature*" e "*model co2*" demonstraram resultados "suficientes", o modelo "*model energy consumption*" demonstrou um resultado "moderado", enquanto "*model water*" demonstrou resultado "fraco".

Tabela 25 – Visão geral classificatória dos melhores gerados para cada algoritmo

Modelo	**FedAVG**	**FedSGD**
model co2	suficiente	suficiente

model energy consumption	fraco	moderado
model temperature	fraco	suficiente
model water	suficiente	fraco

Com isso pode-se responder à questão de pesquisa "**QP2) É possível ao *middleware* proposto ter bom desempenho para os diferentes conjuntos de dados considerados neste trabalho?**". Os dados expostos, demonstram que, embora não tenha sido possível observar de maneira generalista para todos os conjuntos de dados considerados neste trabalho, foi percebido que para os algoritmos distintos de agregação obteve-se bons resultados, "suficientes", para os modelos "*model co2*", "*model temperature*" e "*model water*". Sendo a exceção do modelo "*model energy consumption*", o qual não foi possível em nenhum dos casos observar bons resultados. Desta maneira, foi possível ao *SFMEI*, por meio de distintos algoritmos de agregação, gerar modelos capazes de alcançar bons resultados nos distintos conjuntos de dados considerados neste trabalho.

Ademais, abrem-se oportunidades futuras, para os casos em que não foi possível alcançar um bom desempenho, "suficiente", em ambos os algoritmos. Estas oportunidades abrangem dois aspectos principais, tanto relacionado às características dos conjuntos de dados, quanto a generalização do modelo LSTM utilizado para treinamento. Os conjuntos de dados utilizados podem ser ainda aprimorados, auxiliando na capacidade de predição dos modelos gerados, onde, principalmente, podem ser aplicados pré-processamentos, tratamentos e algoritmos adicionais, citados na Seção 3.3.1, que foram deixados de fora deste estudo (Eldeeb; Alves, 2023). Quanto ao modelo LSTM utilizado, ele foi composto de parâmetros simples, generalistas, aplicados aos distintos conjuntos de dados, sendo possível assim buscar uma otimização deste modelo, ou utilização de modelos com parâmetros específicos para cada conjunto de dados (Kaselimi et al., 2019; Mahmoodzadeh et al., 2022; Ahmed et al., 2023). Ambos os casos podem resultar em um melhor desempenho para o treinamento dos modelos que foram mal classificados.

5.3.2 Quanto ao comparativo entre os algoritmos de agregação

Em uma comparação direta entre os resultados obtidos pela execução dos algoritmos FedAVG e FedSGD pode-se detalhar alguns pontos. Por meio da execução com o algoritmo FedAVG, $\pm$35% dos modelos gerados foram classificados como "suficientes", enquanto que com o algoritmo FedSGD, $\pm$47% dos modelos gerados foram classificados como "suficientes". Da mesma forma, para termos negativos, pode-se também comparar os modelos classificados como "fracos". Por meio da execução com o algoritmo FedAVG, $\pm$59% dos modelos gerados foram classificados como "fracos". Já na execução com o algoritmo FedSGD, estes modelos "fracos" correspondem a $\pm$46% dos modelos gerados.

Além destas classificações, obtidas a partir do valor de R^2, é possível juntamente avaliar a evolução da perda, MSE, para ambos os algoritmos. Por meio do MSE é possível observar

uma maior possibilidade na geração de modelos que tendem a convergir durante a execução do algoritmo FedSGD. Nos modelos gerados por esse algoritmo, é possível notar uma tendência de aproximação à zero em todos os casos, não havendo uma grande variação, ou distanciamento, para nenhum modelo com o passar das épocas e rodadas de treinamento. Assim, é possível responder parcialmente à questão de pesquisa: "**QP3) É possível ao *middleware* proposto ter bom desempenho em algoritmos distintos de aprendizagem federada?**", pois observa-se que para alguns casos ambos os algoritmos tiveram bom desempenho, entretanto, de maneira geral os modelos gerados através do cenário de teste do algoritmo FedSGD obtiveram melhor resultado frente aos modelos gerados através do algoritmo FedAVG.

A Tabela 26 compara os modelos finais agregados pelo *SFMEI*. Estes seriam os modelos disponibilizados para uso através do repositório do *SFMEI*. Baseado nesta tabela e nos resultados expostos até aqui, podemos responder à questão de pesquisa "**QP4) Algum dos algoritmos de agregação possui melhor desempenho que o outro? Quantos porcento?**", pois observando detalhadamente, temos que, pelo valor de R^2, três dos quatro modelos gerados pelo FedSGD se sobressaem aos modelos gerados pelo FedAVG. O modelo "*model co2*" do FedSGD obteve métrica $\pm$13% melhor que o obtido com o FedAVG, já para o modelo "*model temperature*", foi $\pm$30% melhor e para modelo "*model energy consumption*", foi $\pm$41% melhor que o FedAVG. Tornando assim evidente que o algoritmo FedSGD, nos cenários de teste propostos, possui melhor desempenho que o algoritmo FedAVG. Apenas para o modelo "*model water*" o FedAVG se sobressai, pois o modelo gerado pelo FedSGD obteve resultado $\pm$48% pior.

Tabela 26 – Comparação modelos finais por algoritmo agregação

Modelo	**FedAVG R^2**	**FedSGD R^2**	**Diferença Percentual**
model co2	0.773708	**0.911401**	0.137693
model energy consumption	0.341075	**0.645562**	0.304487
model temperature	0.493405	**0.904275**	0.41087
model water	**0.865431**	0.378494	-0.486937

5.3.3 Quanto ao desempenho geral do *SFMEI*

As questões expostas até aqui demonstram que o *SFMEI* é capaz de gerar modelos por meio de aprendizado federado com bom desempenho, mesmo pra distintos conjuntos de dados e distintos algoritmos. E assim, de maneira mais detalhada, o algoritmo FedSGD se sobressai em comparação ao algoritmo FedAVG. Entretanto, conforme os resultados expostos na seção 5.2.5, ocorre um comportamento distinto quando se possui um número reduzido de clientes. Por meio destes resultados é possível observar que o algoritmo FedAVG se sobressai, frente ao algoritmo FedSGD, ao comparar os valores de R^2 obtidos para o modelo "*model water*".

Observando a progressão do valor da perda, MSEé possível verificar um outro aspecto bastante importante. Nelas podem-se observar que em ambas as execuções dos algoritmos, em

todos os modelos gerados por todos os clientes, observa-se um comportamento de convergência de todos os modelos treinados, com MSE aproximando a zero, com o decorrer das épocas.

Assim, respondemos parcialmente a questão de pesquisa "**QP5) Um número reduzido de nós participantes no treinamento pode reduzir a capacidade de um modelo obter um escore alto? Caso sim, quantos porcento?**", pois estes fatores, expostos até o momento, podem ser entendidos como um bom desempenho dos algoritmos empregados no *SFMEI*, no qual, os modelos, ao transitarem entre clientes não alteram de maneira drástica a medida do MSE. Mesmo que, como no caso do algoritmo FedSGD, os modelos tenham sido classificado como "fraco", o comportamento do MSE evidenciam que se aplicam as oportunidades de melhorias citadas na Seção 5.3.1.

Ainda considerando o MSE, podemos observar que apenas para o algoritmo FedSGD ocorre uma aproximação do zero, no decorrer das épocas de treinamento, em todos os casos dos modelos treinados pelos clientes. Neste algoritmo o modelo melhor classificado a cada rodada de federação é repassado para os clientes para as próximas rodadas de aprendizagem. Observando que em todas as rodadas de federação, os melhores modelos foram repassados para todos os clientes, mantendo seu desempenho. Entretanto, mantém-se ainda aberta a possibilidade de uma avaliação de predições em clientes que apenas consumam um modelo pré-treinado pelo *SFMEI*.

5.3.4 Quanto a capacidade de predição de um modelo pré-treinado pelo *SFMEI*

A questão de pesquisa "**QP6) O *middleware* proposto é capaz de fornecer um modelo pré-treinado a ser utilizado para predizer cenários locais sem reduzir drasticamente sua performance?**", pode ser respondida ao analisar os resultados expostos na seção 5.2.7. Através destes resultados é possível validar que o *SFMEI* é capaz de gerar modelos que tendem a se adaptar em distintos clientes com dados distintos. Fator corroborado pelos dados expostos, dos clientes X, Y e Z, presentes na Tabela 24.

Para os clientes X e Z foi possível classificar os modelos utilizados na predição como "suficientes", enquanto no cliente Y o desempenho do modelo foi classificado como "moderado". Entretanto, mesmo que o modelo gerado pelo cliente Y, através da Figura 37 observa-se que os dados previstos mantém-se aproximado dos dados de teste. Sendo assim possível, ao *SFMEI*, de fornecer modelos pré-treinados a serem utilizados sem que reduzam de maneira drástica suas performances, sendo capazes de contribuir bem de forma preditiva, mantendo uma boa performance.

5.4 Ameaças à validade

É necessário ressaltar, diante dos resultados discutidos, a necessidade de alguns avanços no *middleware* desenvolvido, bem como a avaliação e adaptação de outros modelos de aprendizado de máquina, como por exemplo, modelos de classificação. Toda a pesquisa foi realizada

utilizando apenas a geração de modelos de regressão, sem maior capacidade de adaptação. Esta limitação pode ser justificada pela finalidade deste trabalho, entendendo também a viabilidade de se criar uma estrutura minimamente funcional do *middleware* proposto. Como dito, abre-se como uma possibilidade de aperfeiçoamento a adaptação dele para análise de modelos de classificação, utilizando novas abordagens e conjuntos de dados propícios à classificação. Juntamente a isso, abre-se a possibilidade de novas análises, novas métricas e novos resultados obtidos a partir da execução do *middleware*.

Além disso, toda essa pesquisa foi avaliada a partir da execução do *middleware* em um ambiente simulado, através de *containers Docker*, instanciados em uma única rede local, sendo isso um fator limitante. Assim, a validade dele frente a uma implementação real, utilizando máquinas distribuídas, em redes distintas, com dados reais coletados a partir de aplicações reais de *smart campus*, precisa ser avaliada e validada, para verificar se ainda assim o *middleware* é capaz de replicar o cenário avaliado nesta pesquisa.

Por fim, os resultados desta pesquisa também limitam-se a uma estratégia de validação pontual, onde outros aspectos ainda seguem abertos e que não foram considerados neste trabalho. Um destes fatores é a análise do contraste entre o número de parâmetros e modelos gerados pelo FedAVG e FedSGD, os quais podem induzir outros resultados a esta pesquisa. Ademais, o volume de dados trafegados também não foi considerado no escopo deste trabalho.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo geral propor e desenvolver um *middleware* de aprendizado federado para facilitar a integração de soluções de smart campus. O middleware, denominado SFMEI (*Smart Federated Middleware for Educational Institutions*), foi projetado para ser flexível, adaptável e escalável, permitindo a integração de diversos algoritmos de aprendizado de máquina e modelos de dados distribuídos, onde destaca-se ao disponibilizar uma interface para abstrair a lógica de aprendizagem federada. Essa abstração é possível através dos recursos disponibilizados um serviço cliente local, o *SFMEI Local Client*.

A avaliação experimental do SFMEI utilizando os algoritmos FedAVG e FedSGD para a tarefa de regressão com distintos conjuntos de dados demonstrou resultados promissores. O algoritmo FedSGD apresentou melhor desempenho em comparação ao FedAVG, obtendo uma maior proporção de modelos classificados como "suficientes" e um valor mediano de R^2. Além disso, o FedSGD apresentou um padrão de convergência mais consistente, como evidenciado pelas curvas e valores de MSE. Através destes resultados foi possível obter resposta para as questões de pesquisas levantadas.

Quanto aos objetivos específicos, este trabalho foi capaz de alcança-los, quando descreveu o SFMEI proposto, sua arquitetura e suas API's, pública e privada. Além disso, foi desenvolvido o *middleware*, baseado na arquitetura proposta, afim de possibilitar experimentos. Juntamente, foram levantados conjuntos de dados distintos, utilizados em pesquisas realizadas na área de *smart campus*, que foram capazes de ser processados e adaptados para servir como bons dados de testes do SFMEI, possibilitando o treino para modelos. Com isso, foi possível realizar uma avaliação experimental do SFMEI, verificando as métricas coletadas, e também validando a suas funcionalidades.

Como resultado do que foi desenvolvido até o momento, foi publicado e apresentado um artigo no XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia).

6.1 Propostas para Continuação da Pesquisa

Mesmo que os resultados e discussões até aqui demonstrem uma boa performance do *SFMEI*, podemos elencar alguns pontos de melhorias futuras, que vão desde quanto ao tipo de modelos utilizados, a aplicação de dados reais, a adaptações do *SFMEI* e, até mesmo, a possibilidade de tornar o *SFMEI* num produto, ou projeto de código aberto (*open-source*).

6.1.1 Quanto ao tipo de modelo

Neste estudo foi realizada a análise e treinamento de modelos LSTM com foco em predição valores em uma série temporal. Não foi aplicado neste trabalho um modelo de classificação. Por este motivo, abre-se a oportunidade de evolução do SFMEI para compreender o treinamentos de distintos outros tipos de modelos e tarefas.

Uma oportunidade futura considerada é a adaptação do SFMEI para utilizar modelos customizados para cada finalidade de predição. Essa implementação permitiria que o SFMEI aproveitasse as características e vantagens de diferentes modelos de aprendizado de máquina, otimizando o desempenho e performance para os mais distintos cenários. Algumas estratégias podem ser consideradas para implementação destes modelos customizados. Seria possível ajustar a API local do SFMEI Client, permitindo aos usuários definir e integrar seus próprios modelos de aprendizado de máquina. Uma alternativa, mais complexa, seria implementar um mecanismo de seleção de modelo automático que definisse o modelo mais adequado para cada tarefa de predição com base em suas características e requisitos.

6.1.2 Quanto à aplicação de dados reais

Este estudo considerou conjuntos de dados com a finalidade de avaliar o desempenho do SFMEI. Com isso, uma oportunidade futura é a utilização e aplicação de dados reais. Visto o bom desempenho do SFMEI nos conjuntos de dados de teste utilizados, abre-se a oportunidade de aplicar o SFMEI em instituições de ensino reais, utilizando dados coletados localmente para treinar e avaliar os modelos. Estes dados reais de instituições de ensino são cruciais para consolidar a viabilidade e confiabilidade do SFMEI em cenários reais, permitindo identificar áreas de aprimoramento para atender às necessidades específicas do ambiente educacional, contribuindo para o desenvolvimento de soluções neste ambiente. Entretanto, para utilizar dados reais de instituições de ensino, questões éticas precisam ser consideradas, sendo assim necessário estabelecer parcerias com instituições dispostas a fornecer e contribuir com seus dados de forma segura.

6.1.3 Quanto aos algoritmos de agregação

Abre-se a oportunidade do SFMEI possibilitar a utilização simultânea de distintos algoritmos de agregação. Isto poderia permitir ao SFMEI usufruir das características e vantagens de diferentes algoritmos, otimizando o desempenho para diferentes tipos de dados e tarefas, permitindo a seleção do algoritmo mais adequado para cada tarefa de predição. Além disso, possibilitaria a exploração de novos algoritmos de agregação emergentes, que podem apresentar melhor desempenho em cenários específicos. Para isso, poderia ser utilizada uma abordagem de aprendizagem por reforço para otimizar a seleção do algoritmo de agregação, além da implementação de um mecanismo de "meta-aprendizagem"adaptável (Nagabandi et al., 2018),

com o intuito de aprender a escolher o algoritmo mais adequado com base em dados históricos e métricas de desempenho.

6.1.4 Quanto à adaptações do *SFMEI*

6.1.4.1 Adaptação para pontuar participantes do treinamento

Uma proposta interessante seria implementar um sistema de pontuação para colaboradores do aprendizado federado no SFMEI. Essa iniciativa visa estimular o compartilhamento de modelos e evitar um possível "efeito carona", no qual algumas instituições apenas consomem modelos sem contribuir com seus próprios dados e modelos. Para implementar esta adaptação poderia ser utilizado como inspiração o sistema do BitTorrent, em que a pontuação poderia ser atribuída de acordo com a frequência e qualidade das contribuições de cada membro (Roughgarden, 2016). Isso incentivaria a participação ativa e recompensaria os membros que mais contribuem para o aprendizado federado do SFMEI. Além disso, outra oportunidade seria desenvolver uma avaliação da qualidade da contribuição por cliente, de modo que modelos sejam considerados para agregação em relação aos clientes que possuam melhores pontuações, ou melhor métrica de avaliação.

6.1.4.2 Adaptação para um aprendizado federado contínuo

Uma oportunidade futura promissora reside na adaptação do SFMEI para implementar um processo de aprendizado federado contínuo, podendo contribuir diretamente para estudos relacionados a AutoFL (Preuveneers, 2023). Essa implementação permitiria que o modelo fosse atualizado e aprimorado continuamente, à medida que novos dados se tornam disponíveis, sem a necessidade de recomeçar o processo de treinamento completo. Isso traria diversos benefícios, permitindo que o modelo se adaptasse às mudanças nas distribuições dos dados ao longo do tempo, garantindo sua precisão a distintos cenários. Além disso, poderia possibilitar a atualização do modelo em tempo real, permitindo que ele responda rapidamente a eventos e mudanças no ambiente.

6.1.4.3 Adaptação para implementar técnicas de privacidade e segurança

Uma proposta futura seria incrementar ao *SFMEI* aspectos de segurança. Embora os dados locais dos nós não sejam trafegados para o servidor central de agregação durante o treinamento de modelos, protocolos e algoritmos de proteção de dados podem ser utilizados para aprimorar ainda mais a segurança do *SFMEI* e dos dados trafegados e mantidos localmente, ou via repositório. Bem como um controle de acesso através de autorização e autenticação, para mediar a participação e acoplamento de instituições de ensindo ao *SFMEI*.

6.1.5 Quanto a tornar o *SFMEI* como um produto, ou projeto *open-source*

Tornar o *SFMEI* num produto abriria oportunidades de mercado (Shi et al., 2021) para uso em cenários de ambientes privados. Desta maneira seria permitido o uso para aplicações críticas em que o vazamento de dados é inaceitável, bem como para a colaboração entre diferentes empresas e instituições, sem a necessidade de compartilhamento de dados entre si.

Tornar o *SFMEI* num projeto *open-source* possibilitaria uma evolução participativa e colaborativa. Projetos *open-source* permitem que uma comunidade, um grupo de desenvolvedores, contribuam para o desenvolvimento e evolução destes projetos. Desta maneira estes projetos podem evoluir a partir das demandas da comunidade.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, A. A.; RAMADHAN, F. A. A.; ELATAB, S. A. Internet of things (iot) middleware a comprehensive overview. 2022. Citado 2 vezes nas páginas 21 e 30.

AHMED, A. N. et al. Forecasting of fine particulate matter based on lstm and optimization algorithm. **Journal of Cleaner Production**, Elsevier, v. 427, p. 139233, 2023. Citado na página 92.

AHMED, E. et al. The role of big data analytics in internet of things. **Computer Networks**, Elsevier, v. 129, p. 459–471, 2017. Citado na página 17.

AÏVODJI, U. M.; GAMBS, S.; MARTIN, A. Iotfla: A secured and privacy-preserving smart home architecture implementing federated learning. In: **2019 IEEE Security and Privacy Workshops (SPW)**. [S.l.: s.n.], 2019. p. 175–180. Citado na página 16.

AL-GARADI, M. A. et al. A survey of machine and deep learning methods for internet of things (iot) security. **IEEE Communications Surveys & Tutorials**, IEEE, v. 22, n. 3, p. 1646–1685, 2020. Citado 2 vezes nas páginas 17 e 18.

ALAM, T. et al. Smart campus mobile application toward the development of smart cities. **Tanweer Alam. Yazeed Mohammed Alharbi. Firas Adel Abusallama. Ahmad Osama Hakeem. Smart Campus Mobile Application Toward the Development of Smart Cities. International Journal of Applied Sciences and Smart Technologies. Vol2**, n. 1, 2020. Citado 2 vezes nas páginas 18 e 31.

ALHAJ, A. et al. Improving the smart cities traffic management systems using vanets and iot features. **Journal of Statistics Applications & Probability**, v. 12, n. 2, p. 405–414, 2023. Citado na página 64.

ARAFEH, M. et al. Modularfed: Leveraging modularity in federated learning frameworks. **Internet of Things**, Elsevier, v. 22, p. 100694, 2023. Citado 2 vezes nas páginas 22 e 112.

AUGUSTO, J. C. A smart campus template. In: **Intelligent Environments 2021**. [S.l.]: IOS Press, 2021. p. 71–80. Citado 2 vezes nas páginas 18 e 21.

AWS. **What's the Difference Between MariaDB and MySQL?** 2023. Https://aws.amazon.com/compare/the-difference-between-mariadb-vs-mysql/. Citado na página 46.

AYDIN, B.; SARI, T. T.; OKTUG, S. F. Accelerating smart campus development with an extensible framework. **IEEE Potentials**, v. 42, n. 4, p. 24–28, 2023. Citado 3 vezes nas páginas 18, 20 e 111.

BANDYOPADHYAY, S. et al. Role of middleware for internet of things: A study. **International Journal of Computer Science and Engineering Survey**, Academy & Industry Research Collaboration Center(AIRCC), v. 2, n. 3, p. 94–105, 2011. Citado na página 30.

BEHERA, M. R. et al. Federated learning using distributed messaging with entitlements for anonymous computation and secure delivery of model. TechRxiv, 2020. Citado na página 18.

BELTRÁN, E. T. M. et al. Fedstellar: A platform for decentralized federated learning. **arXiv e-prints**, p. arXiv–2306, 2023. Citado na página 28.

BENNIS, M.; DEBBAH, M.; POOR, H. V. Ultrareliable and low-latency wireless communication: Tail, risk, and scale. **Proceedings of the IEEE**, v. 106, n. 10, p. 1834–1853, 2018. Citado na página 16.

BEUTEL, D. J. et al. Flower: A friendly federated learning framework. 2022. Citado 2 vezes nas páginas 22 e 111.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4. Citado na página 45.

BLAZEVIC, M.; RIEHLE, D. M. University of things: Opportunities and challenges for a smart campus environment based on iot sensors and business processes. **IoTBDS**, p. 105–114, 2023. Citado na página 37.

BOLBOACĂ, R.; HALLER, P. Performance analysis of long short-term memory predictive neural networks on time series data. **Mathematics**, MDPI, v. 11, n. 6, p. 1432, 2023. Citado na página 48.

BONAWITZ, K. et al. Towards federated learning at scale: System design. **Proceedings of machine learning and systems**, v. 1, p. 374–388, 2019. Citado na página 18.

BRAND, B. S. et al. Sapientia: a smart campus model to promote device and application flexibility. **Advances in Computational Intelligence**, Springer, v. 2, n. 1, p. 1–10, 2022. Citado 2 vezes nas páginas 18 e 31.

BRIK, B.; KSENTINI, A.; BOUAZIZ, M. Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems. p. 53841–53849, 2020. Citado na página 16.

BRUCE, A.; BRUCE, P. **Estatística prática para cientistas de dados**. [S.l.]: Alta Books, 2019. Citado 2 vezes nas páginas 41 e 45.

CAO, H. et al. Ifed: A novel federated learning framework for local differential privacy in power internet of things. **International Journal of Distributed Sensor Networks**, SAGE Publications Sage UK: London, England, v. 16, n. 5, p. 1550147720919698, 2020. Citado 2 vezes nas páginas 16 e 29.

CAVUS, N. et al. Internet of things and its applications to smart campus: A systematic literature review. **International Journal of Interactive Mobile Technologies**, v. 17, n. 23, 2022. Citado na página 25.

CHAGNON-LESSARD, N. et al. Smart campuses: extensive review of the last decade of research and current challenges. **IEEE Access**, IEEE, v. 9, p. 124200–124234, 2021. Citado 9 vezes nas páginas 17, 18, 20, 25, 30, 31, 32, 33 e 38.

CHEN, C. Research on online teaching emotion detection based on federated learning. In: **Proceedings of the 2023 3rd International Conference on Bioinformatics and Intelligent Computing**. [S.l.: s.n.], 2023. p. 159–164. Citado na página 32.

CHICCO, D.; WARRENS, M. J.; JURMAN, G. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. **Peerj computer science**, PeerJ Inc., v. 7, p. e623, 2021. Citado 3 vezes nas páginas 43, 44 e 45.

COHEN, J. et al. **Applied multiple regression/correlation analysis for the behavioral sciences**. [S.l.]: Routledge, 2013. Citado 2 vezes nas páginas 44 e 45.

CORRADINI, F. et al. Floware: a model-driven approach fostering reuse and customisation in iot applications modelling and development. **Software and Systems Modeling**, Springer, v. 22, n. 1, p. 131–158, 2023. Citado 2 vezes nas páginas 20 e 111.

DAMASKINOS, G. et al. Fleet: Online federated learning via staleness awareness and performance prediction. In: **Proceedings of the 21st International Middleware Conference**. [S.l.: s.n.], 2020. p. 163–177. Citado 3 vezes nas páginas 22, 33 e 112.

DASARI, S. V. et al. Privacy enhanced energy prediction in smart building using federated learning. In: **2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)**. [S.l.: s.n.], 2021. Citado na página 16.

DRESCH, A.; LACERDA, D. P.; JUNIOR, J. A. V. A. **Design science research: método de pesquisa para avanço da ciência e tecnologia**. [S.l.]: Bookman Editora, 2020. Citado na página 37.

ELDEEB, E.; ALVES, H. Lorawan-enabled smart campus: The dataset and a people counter use case. **arXiv preprint arXiv:2304.13366**, 2023. Citado 7 vezes nas páginas 19, 20, 40, 41, 45, 92 e 111.

FREIRE, L. d. P. Gerenciamento de informações no contexto de smart campus: uma revisão sistemática da literatura. Centro Multidisciplinar de Pau dos Ferros, 2023. Citado na página 18.

FUSHIKI, T. Estimation of prediction error by using k-fold cross-validation. **Statistics and Computing**, Springer, v. 21, p. 137–146, 2011. Citado na página 48.

GAO, D. et al. Fs-real: A real-world cross-device federated learning platform. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 16, n. 12, p. 4046–4049, 2023. Citado na página 28.

GIURIATTI, T. et al. Smart campus: Uma revisão da literatura para identificação das dimensões que compreendem a customização universitária. In: **Anais do Congresso Internacional de Conhecimento e Inovação–ciki**. [S.l.: s.n.], 2023. Citado 3 vezes nas páginas 25, 32 e 38.

GORE, S. et al. Innovations in smart city water supply systems. **International Journal of Intelligent Systems and Applications in Engineering**, v. 11, n. 9s, p. 277–281, 2023. Citado na página 64.

GOUGH, J.; BRYANT, D.; AUBURN, M. **Mastering API Architecture**. [S.l.]: "O'Reilly Media, Inc.", 2021. Citado 2 vezes nas páginas 54 e 65.

GUO, S.; ZENG, D.; DONG, S. Pedagogical data analysis via federated learning toward education 4.0. **American Journal of Education and Information Technology**, Science Publishing Group, v. 4, p. 56, 2020. Citado 3 vezes nas páginas 18, 32 e 111.

GUO, Y. Application of internet of things technology in mobile education of smart campus culture and etiquette. **Journal of Sensors**, Hindawi, v. 2022, 2022. Citado 2 vezes nas páginas 18 e 31.

HAIR, J. F. et al. When to use and how to report the results of pls-sem. **European business review**, Emerald Publishing Limited, v. 31, n. 1, p. 2–24, 2019. Citado 4 vezes nas páginas 44, 48, 49 e 68.

HAMER, J.; MOHRI, M.; SURESH, A. T. Fedboost: A communication-efficient algorithm for federated learning. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2020. p. 3973–3983. Citado na página 28.

HARRISON, M. **Machine learning pocket reference: working with structured data in python**. [S.l.]: O'Reilly Media, 2019. Citado 3 vezes nas páginas 20, 41 e 45.

HASSAN, A. et al. Sentiment analysis on bangla and romanized bangla text using deep recurrent models. In: IEEE. **2016 International Workshop on Computational Intelligence (IWCI)**. [S.l.], 2016. p. 51–56. Citado 2 vezes nas páginas 44 e 45.

HE, C. et al. Fedcv: a federated learning framework for diverse computer vision tasks. **arXiv preprint arXiv:2111.11066**, 2021. Citado 2 vezes nas páginas 22 e 111.

HEY, T.; TREFETHEN, A. The data deluge: An e-science perspective. **Grid computing: Making the global infrastructure a reality**, Wiley Online Library, p. 809–824, 2003. Citado 2 vezes nas páginas 16 e 17.

HUANG, C.; HUANG, J.; LIU, X. Cross-silo federated learning: Challenges and opportunities. **arXiv preprint arXiv:2206.12949**, 2022. Citado na página 28.

IDC. **Business Models for the Long-Term Storage of Internet of Things Use Case Data**. 2020. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=AP45984120>. Citado na página 16.

IDC. **Data Age 2025: The Evolution of Data to Life-Critical. Don't Focus on Big Data; Focus on the Data That's Big**. 2022. Disponível em: <https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf>. Citado na página 16.

IMTEAJ, A. et al. A survey on federated learning for resource-constrained iot devices. **IEEE Internet of Things Journal**, IEEE, v. 9, n. 1, p. 1–24, 2021. Citado na página 28.

JAMES, G. et al. **An introduction to statistical learning**. [S.l.]: Springer, 2013. v. 112. Citado 3 vezes nas páginas 43, 45 e 48.

KAIROUZ, P. et al. Advances and open problems in federated learning. **CoRR**, abs/1912.04977, 2019. Disponível em: <http://arxiv.org/abs/1912.04977>. Citado 2 vezes nas páginas 18 e 25.

KASELIMI, M. et al. Bayesian-optimized bidirectional lstm regression model for non-intrusive load monitoring. In: IEEE. **ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2019. p. 2747–2751. Citado na página 92.

KAYMAKCI, C.; BAUR, L.; SAUER, A. Federated machine learning architecture for energy-efficient industrial applications. **ESSN: 2701-6277**, Hannover: Institutionelles Repositorium der Leibniz Universität Hannover, 2021. Citado na página 18.

KHAN, A. N. et al. Hetero-fediot: A rule-based interworking architecture for heterogeneous federated iot networks. **IEEE Internet of Things Journal**, IEEE, 2023. Citado 2 vezes nas páginas 22 e 112.

KHAN, L. U. et al. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. 2021. Citado 2 vezes nas páginas 18 e 112.

KIM, Y. G.; WU, C.-J. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In: **MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture**. [S.l.: s.n.], 2021. p. 183–198. Citado na página 17.

KONEČNỲ, J. et al. Federated optimization: Distributed machine learning for on-device intelligence. **arXiv preprint arXiv:1610.02527**, 2016. Citado 2 vezes nas páginas 18 e 25.

KONTAR, R. et al. The internet of federated things (ioft): A vision for the future and in-depth survey of data-driven approaches for federated learning. **arXiv preprint arXiv:2111.05326**, 2021. Citado 2 vezes nas páginas 27 e 62.

LI, D.; WANG, J. Fedmd: Heterogenous federated learning via model distillation. **arXiv preprint arXiv:1910.03581**, 2019. Citado na página 17.

LI, T. et al. Federated optimization in heterogeneous networks. **Proceedings of Machine learning and systems**, v. 2, p. 429–450, 2020. Citado na página 17.

LI, X. et al. Context aware middleware architectures: Survey and challenges. **Sensors**, MDPI, v. 15, n. 8, p. 20570–20607, 2015. Citado na página 30.

LIM, W. Y. B. et al. Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach. **IEEE Transactions on Intelligent Transportation Systems**, 2021. Citado na página 30.

LIU, Y. et al. A secure federated learning framework for 5g networks. **IEEE Wireless Communications**, IEEE, v. 27, n. 4, p. 24–31, 2020. Citado 2 vezes nas páginas 22 e 111.

LUPTON, B. et al. Analysis and prevention of security vulnerabilities in a smart city. In: IEEE. **2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.], 2022. p. 0702–0708. Citado na página 64.

LV, Z. The integrated processing method of educational information resources based on edge computing. **Journal of Electrical and Computer Engineering**, Hindawi, v. 2022, 2022. Citado 2 vezes nas páginas 18 e 31.

MAHMOODZADEH, A. et al. Forecasting tunnel boring machine penetration rate using lstm deep neural network optimized by grey wolf optimization algorithm. **Expert Systems with Applications**, Elsevier, v. 209, p. 118303, 2022. Citado na página 92.

MAKONIN, S. **ODDs: Occupancy Detection Dataset**. Harvard Dataverse, 2015. Disponível em: <https://doi.org/10.7910/DVN/2K9FFE>. Citado na página 40.

MAKONIN, S. **AMPds2: The Almanac of Minutely Power dataset (Version 2)**. Harvard Dataverse, 2016. Disponível em: <https://doi.org/10.7910/DVN/FIE0S4>. Citado na página 40.

MAKONIN, S. et al. Electricity, water, and natural gas consumption of a residential house in canada from 2012 to 2014. **Scientific data**, Nature Publishing Group, v. 3, n. 1, p. 1–12, 2016. Citado 2 vezes nas páginas 19 e 40.

MARIADB. **MariaDB Licenses**. 2024. Https://mariadb.com/kb/en/mariadb-licenses/. Citado na página 64.

MAZZOCCA, C. et al. Framh: A federated learning risk-based authorization middleware for healthcare. **IEEE Transactions on Computational Social Systems**, IEEE, 2022. Citado 2 vezes nas páginas 22 e 112.

MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. **Artificial intelligence and statistics**. [S.l.], 2017. p. 1273–1282. Citado na página 27.

MITRA, A.; NGOKO, Y.; TRYSTRAM, D. Impact of federated learning on smart buildings. In: **2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)**. [S.l.: s.n.], 2021. p. 93–99. Citado na página 16.

MOCANU, E. et al. Deep learning for estimating building energy consumption. **Sustainable Energy, Grids and Networks**, v. 6, p. 91–99, 2016. Citado 2 vezes nas páginas 16 e 18.

MOHRI, M.; SIVEK, G.; SURESH, A. T. Agnostic federated learning. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2019. p. 4615–4625. Citado 2 vezes nas páginas 22 e 112.

MONTI, L. et al. Edge-based transfer learning for classroom occupancy detection in a smart campus context. **Sensors**, MDPI, v. 22, n. 10, p. 3692, 2022. Citado 2 vezes nas páginas 18 e 31.

MORALIYAGE, H. et al. **UNICON Energy Consumption Dataset**. 2018. Https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption. Citado na página 40.

MORALIYAGE, H. et al. Unicon: An open dataset of electricity, gas and water consumption in a large multi-campus university setting. In: IEEE. **2022 15th International Conference on Human System Interaction (HSI)**. [S.l.], 2022. p. 1–8. Citado 2 vezes nas páginas 19 e 40.

MUHAMAD, W. et al. Smart campus features, technologies, and applications: A systematic literature review. In: IEEE. **2017 International conference on information technology systems and innovation (ICITSI)**. [S.l.], 2017. p. 384–391. Citado na página 18.

MYSQL. **Commercial License for OEMs, ISVs and VARs**. 2010. Https://www.mysql.com/about/legal/licensing/oem/. Citado na página 64.

NAGABANDI, A. et al. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. **arXiv preprint arXiv:1803.11347**, 2018. Citado na página 97.

NAKAGAWA, S.; SCHIELZETH, H. A general and simple method for obtaining r2 from generalized linear mixed-effects models. **Methods in ecology and evolution**, Wiley Online Library, v. 4, n. 2, p. 133–142, 2013. Citado 2 vezes nas páginas 44 e 45.

NASIRIFARD, P.; MAYER, R.; JACOBSEN, H.-A. Orderlessfl: a crdt-enabled permissioned blockchain for federated learning. In: **Proceedings of the 23rd International Middleware Conference Demos and Posters**. [S.l.: s.n.], 2022. p. 7–8. Citado na página 22.

NETER, J. et al. Applied linear statistical models. Irwin Chicago, 1996. Citado na página 69.

NETTO, R. S. et al. Facens smart campus integrated dashboard: A use case applied for energy efficiency. In: **IoT and WSN based Smart Cities: A Machine Learning Perspective**. [S.l.]: Springer, 2022. p. 67–88. Citado 3 vezes nas páginas 41, 43 e 44.

NGUYEN, D. C. et al. Federated learning for internet of things: A comprehensive survey. **arXiv preprint arXiv:2104.07914**, 2021. Citado 3 vezes nas páginas 18, 25 e 29.

NGUYEN, H. T. et al. Fast-convergent federated learning. **IEEE Journal on Selected Areas in Communications**, IEEE, v. 39, n. 1, p. 201–218, 2020. Citado na página 28.

NÓBREGA, P. I. Silva-da; CHIM-MIKI, A. F.; CASTILLO-PALACIO, M. A smart campus framework: Challenges and opportunities for education based on the sustainable development goals. **Sustainability**, MDPI, v. 14, n. 15, p. 9640, 2022. Citado 2 vezes nas páginas 32 e 38.

OULU, U. of. **Smart Campus Oulu indoor climate, air-quality and motion**. 2023. <https://doi.org/10.23729/b9adb0a2-7381-45db-b32f-7e78ae1bc9e3>. University of Oulu, CWC - Verkot ja järjestelmät. Citado 2 vezes nas páginas 40 e 88.

PEFFERS, K. et al. A design science research methodology for information systems research. **Journal of management information systems**, JSTOR, p. 45–77, 2007. Citado na página 37.

PIMENTEL, M.; FILIPPO, D.; SANTORO, F. M. Design science research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação. **Metodologia de Pesquisa em Informática na Educação: Concepção da Pesquisa. Porto Alegre: SBC**, 2019. Citado na página 37.

POORNIMA, I. G. A.; PARAMASIVAN, B. Anomaly detection in wireless sensor network using machine learning algorithm. **Computer Communications**, Elsevier, v. 151, p. 331–337, 2020. Citado na página 17.

PREUVENEERS, D. Autofl: Towards automl in a federated learning context. **Applied Sciences**, MDPI, v. 13, n. 14, p. 8019, 2023. Citado na página 98.

PROJECT, D. **Reasons to use Debian**. 2024. Https://www.debian.org/intro/why$_{d}$*ebian.en.html*.*Citadonapgina*64.

QI, P. et al. Model aggregation techniques in federated learning: A comprehensive survey. **Future Generation Computer Systems**, Elsevier, 2023. Citado na página 28.

RADY, H. A. K. Shannon entropy and mean square errors for speeding the convergence of multilayer neural networks: A comparative approach. **Egyptian Informatics Journal**, Elsevier, v. 12, n. 3, p. 197–209, 2011. Citado na página 69.

RAZZAQUE, M. A. et al. Middleware for internet of things: a survey. **IEEE Internet of things journal**, IEEE, v. 3, n. 1, p. 70–95, 2015. Citado na página 30.

REHMAN, M. H. ur et al. Trustfed: A framework for fair and trustworthy cross-device federated learning in iiot. **IEEE Transactions on Industrial Informatics**, IEEE, v. 17, n. 12, p. 8485–8494, 2021. Citado na página 28.

RIEKE, N. et al. The future of digital health with federated learning. **NPJ digital medicine**, Nature Publishing Group, v. 3, p. 1–7, 2020. Citado na página 29.

RIGDON, E. E. Rethinking partial least squares path modeling: In praise of simple methods. **Long range planning**, Elsevier, v. 45, n. 5-6, p. 341–358, 2012. Citado na página 44.

ROMANINI, D. et al. Pyvertical: A vertical federated learning framework for multi-headed splitnn. **arXiv preprint arXiv:2104.00489**, 2021. Citado 2 vezes nas páginas 22 e 111.

ROUGHGARDEN, T. Cs269i: Incentives in computer science lecture# 5: Incentives in peer-to-peer networks. 2016. Citado na página 98.

SANTOS, L. E. B. D. et al. Middleware for smart campus applications based in federated learning. In: **Proceedings of the Brazilian Symposium on Multimedia and the Web**. [S.l.: s.n.], 2022. p. 324–328. Citado 2 vezes nas páginas 46 e 71.

SATER, R. A.; HAMZA, A. B. A federated learning approach to anomaly detection in smart buildings. **ACM Transactions on Internet of Things**, ACM New York, NY, USA, v. 2, n. 4, p. 1–23, 2021. Citado na página 16.

SEDJELMACI, H.; SENOUCI, S. M.; AL-BAHRI, M. A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology. In: IEEE. **2016 IEEE international conference on communications (ICC)**. [S.l.], 2016. p. 1–6. Citado na página 16.

SHAER, I.; SHAMI, A. Corrfl: Correlation-based neural network architecture for unavailability concerns in a heterogeneous iot environment. **IEEE Transactions on Network and Service Management**, v. 20, n. 2, p. 1543–1557, 2023. Citado 3 vezes nas páginas 17, 22 e 111.

SHAH, R. et al. Linear regression vs lstm for time series data. In: IEEE. **2022 IEEE World Conference on Applied Intelligence and Computing (AIC)**. [S.l.], 2022. p. 670–675. Citado na página 48.

SHEN, M. et al. Nonlinear hyperparameter optimization of a neural network in image processing for micromachines. **Micromachines**, MDPI, v. 12, n. 12, p. 1504, 2021. Citado na página 69.

SHI, Y. et al. User privacy leakages from federated learning in nilm applications. In: **Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation**. [S.l.: s.n.], 2021. p. 212–213. Citado 3 vezes nas páginas 17, 25 e 99.

SO, J. et al. Fedspace: An efficient federated learning framework at satellites and ground stations. **arXiv preprint arXiv:2202.01267**, 2022. Citado 2 vezes nas páginas 22 e 112.

STRECHT, P. et al. A comparative study of classification and regression algorithms for modelling students' academic performance. **International educational data mining society**, ERIC, 2015. Citado na página 42.

SUN, W. Predictive analysis and simulation of college sports performance fused with adaptive federated deep learning algorithm. **Journal of Sensors**, Hindawi, v. 2022, 2022. Citado na página 32.

TAÏK, A.; CHERKAOUI, S. Electrical load forecasting using edge computing and federated learning. In: **ICC 2020-2020 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2020. Citado na página 16.

TAN, L.; WANG, N. Future internet: The internet of things. In: **2010 3rd international conference on advanced computer theory and engineering (ICACTE)**. [S.l.: s.n.], 2010. v. 5, p. V5–376. Citado na página 16.

TERRAIL, J. O. d. et al. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. **arXiv preprint arXiv:2210.04620**, 2022. Citado na página 28.

VEMURI, V. K. **The Hundred-Page Machine Learning Book: by Andriy Burkov, Quebec City, Canada, 2019, 160 pp., 49.99(Hardcover); 29.00 (paperback); 25.43(KindleEdition),(Alternatively,canpurchaseatleanpub.comataminimumpriceof 20.00), ISBN 978-1999579517**. [S.l.]: Taylor & Francis, 2019. Citado na página 17.

VERBRAEKEN, J.; VOS, M. de; POUWELSE, J. Bristle: Decentralized federated learning in byzantine, non-iid environments. **arXiv preprint arXiv:2110.11006**, 2021. Citado 3 vezes nas páginas 22, 33 e 112.

WANG, J. et al. A novel framework for the analysis and design of heterogeneous federated learning. **IEEE Transactions on Signal Processing**, IEEE, v. 69, p. 5234–5249, 2021. Citado na página 17.

WENG, Y.; ZHANG, N.; XIA, C. Multi-agent-based unsupervised detection of energy consumption anomalies on smart campus. **IEEE Access**, IEEE, v. 7, p. 2169–2178, 2018. Citado 2 vezes nas páginas 19 e 40.

WONG, T.-T.; YEH, P.-Y. Reliable accuracy estimates from k-fold cross validation. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 32, n. 8, p. 1586–1594, 2019. Citado na página 48.

WU, H. et al. Research on artificial intelligence enhancing internet of things security: A survey. **Ieee Access**, v. 8, p. 153826–153848, 2020. Citado na página 18.

WU, H.; WANG, P. Fast-convergent federated learning with adaptive weighting. **IEEE Transactions on Cognitive Communications and Networking**, IEEE, v. 7, n. 4, p. 1078–1088, 2021. Citado na página 28.

WU, X. et al. Faster adaptive federated learning. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2023. v. 37, n. 9, p. 10379–10387. Citado na página 28.

WU, X.; LIANG, Z.; WANG, J. Fedmed: A federated learning framework for language modeling. **Sensors**, MDPI, v. 20, n. 14, p. 4048, 2020. Citado 2 vezes nas páginas 22 e 112.

XU, J. et al. Federated learning for healthcare informatics. **Journal of Healthcare Informatics Research**, Springer, p. 1–19, 2021. Citado 2 vezes nas páginas 22 e 112.

YIN, F. et al. Fedloc: Federated learning framework for data-driven cooperative localization and location data processing. **IEEE Open Journal of Signal Processing**, IEEE, v. 1, p. 187–215, 2020. Citado 2 vezes nas páginas 22 e 112.

ZANELLA, A. et al. Internet of things for smart cities. **IEEE Internet of Things journal**, v. 1, p. 22–32, 2014. Citado na página 16.

ZENG, D. et al. Fedlab: A flexible federated learning framework. **Journal of Machine Learning Research**, v. 24, n. 100, p. 1–7, 2023. Citado 3 vezes nas páginas 22, 28 e 111.

ZHANG, J. et al. Fedala: Adaptive local aggregation for personalized federated learning. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2023. v. 37, n. 9, p. 11237–11244. Citado na página 28.

ZHANG, S. et al. Improving the accuracy of load forecasting for campus buildings based on federated learning. In: **2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)**. [S.l.: s.n.], 2021. p. 1–5. Citado 2 vezes nas páginas 32 e 33.

ZHANG, T. et al. Federated learning for internet of things: a federated learning framework for on-device anomaly data detection. **arXiv preprint arXiv:2106.07976**, 2021. Citado 6 vezes nas páginas 18, 22, 33, 35, 39 e 46.

ZHANG, X. et al. Fedpd: A federated learning framework with adaptivity to non-iid data. **IEEE Transactions on Signal Processing**, IEEE, v. 69, p. 6055–6070, 2021. Citado 2 vezes nas páginas 22 e 111.

ZHANG, X.; LIU, C.-A. Model averaging prediction by k-fold cross-validation. **Journal of Econometrics**, Elsevier, v. 235, n. 1, p. 280–301, 2023. Citado na página 48.

ZHANG, Y. et al. A systematic review on technologies and applications in smart campus: A human-centered case study. **IEEE Access**, v. 10, p. 16134–16149, 2022. Citado na página 38.

ZHOU, W. et al. Real-time data processing architecture for multi-robots based on differential federated learning. In: **2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)**. [S.l.: s.n.], 2018. p. 462–471. Citado na página 29.

Apêndices

APÊNDICE A – TABELA DE REFERENCIA DE ESTUDOS ANALISADOS EM *SMART CAMPUS* E APRENDIZADO FEDERADO

Tabela 27: Tabela de referencia de estudos analisados em *Smart Campus* e aprendizado federado

Estudo	Nome	Chave Referencial	Tipo
1	*Pedagogical Data Analysis Via Federated Learning Toward Education 4.0* (Guo; Zeng; Dong, 2020)	*PedagogicalFL4.0*	*Smart Campus*
2	*Accelerating Smart Campus Development with an Extensible Framework* (Aydin; Sari; Oktug, 2023)	*SmartCampusFramework*	*Smart Campus*
3	*LoRaWAN-enabled Smart Campus: The Dataset and a People Counter Use Case* (Eldeeb; Alves, 2023)	*LoRaWan-SmartCampus*	*Smart Campus*
4	*FloWare: a model-driven approach fostering reuse and customisation in IoT applications modelling and development* (Corradini et al., 2023)	*FloWare*	Aprendizado Federado
5	*CorrFL: Correlation-Based Neural Network Architecture for Unavailability Concerns in a Heterogeneous IoT Environment* (Shaer; Shami, 2023)	*CorrFL*	Aprendizado Federado
6	*A secure federated learning framework for 5G networks* (Liu et al., 2020)	*SecureFL5.0*	Aprendizado Federado
7	*Fedlab: A flexible federated learning framework* (Zeng et al., 2023)	*FedLab*	Aprendizado Federado
8	*Flower: A friendly federated learning framework* (Beutel et al., 2022)	*Flower*	Aprendizado Federado
9	*FedPD: A federated learning framework with adaptivity to non-iid data* (Zhang et al., 2021c)	*FedPD*	Aprendizado Federado
10	*Pyvertical: A vertical federated learning framework for multi-headed splitnn* (Romanini et al., 2021)	*Pyvertical*	Aprendizado Federado
11	*Fedcv: a federated learning framework for diverse computer vision tasks* (He et al., 2021)	*Fedcv*	Aprendizado Federado

Continued on next page

Tabela 27: Tabela de referencia de estudos analisados em *Smart Campus* e aprendizado federado (Continued)

12	*Agnostic federated learning* (Mohri; Sivek; Suresh, 2019)	*AgnosticFL*	Aprendizado Federado
13	*FedLoc: Federated learning framework for data-driven cooperative localization and location data processing* (Yin et al., 2020)	*FedLoc*	Aprendizado Federado
14	*Fedmed: A federated learning framework for language modeling* (Wu; Liang; Wang, 2020)	*Fedmed*	Aprendizado Federado
15	*Federated learning for healthcare informatics* (Xu et al., 2021)	*FLHealthcare*	Aprendizado Federado
16	*Fedspace: An efficient federated learning framework at satellites and ground stations* (So et al., 2022)	*Fedspace*	Aprendizado Federado
17	*Hetero-FedIoT: A Rule-Based Interworking Architecture for Heterogeneous Federated IoT Networks* (Khan et al., 2023)	*Hetero-FedIot*	Aprendizado Federado
18	*OrderlessFL: a CRDT-enabled permissioned blockchain for federated learning*	*OrderlessFL*	Aprendizado Federado
19	*ModularFed: Leveraging modularity in federated learning frameworks* (Arafeh et al., 2023)	*ModularFed*	Aprendizado Federado
20	*FRAMH: A Federated Learning Risk-Based Authorization Middleware for Healthcare* (Mazzocca et al., 2022)	*FRAMH*	Aprendizado Federado
21	*Fleet: Online federated learning via staleness awareness and performance prediction* (Damaskinos et al., 2020)	*Fleet*	Aprendizado Federado
22	*Bristle: Decentralized Federated Learning in Byzantine, Non-iid Environments* (Verbraeken; Vos; Pouwelse, 2021)	*Bristle*	Aprendizado Federado
23	*Federated learning for internet of things: a federated learning framework for On-device anomaly data detection* (Khan et al., 2021)	*FLIoT*	Aprendizado Federado

APÊNDICE B – TABELAS DA RELAÇÃO DE ESTUDOS DE APRENDIZADO FEDERADO EM *SMART CAMPUS*

Tabela 28 – Relação de estudos aprendizado federado com categorias de estudos em *smart campus* - Parte 1 - Estudos 1 a 12

SMART CAMPUS CATEGORIES	SFMEI	STUDIES											
		1	2	3	4	5	6	7	8	9	10	11	12
Analyses Of Current Building Stock And Building Refurbishment										X	X		
Big Data Platforms And Frameworks		X	X	X	X	X	X	X	X			X	
Integration Of Real Time Data Collection Into Bim Environment			X		X			X		X	X	X	
Artificial Intelligence For Pattern Recognition Fault Detection And Control	X					X	X	X	X	X	X		X
Role Of Occupants				X							X		
Microgrids			X							X			
Campus Scale Sustainability Initiatives	X											X	X
Waste Management And Recycling	X												
Campus Environmental Monitoring	X								X	X			
Monitoring And Understanding Mobility On Campuses			X	X					X	X			
Bus Transportation													
Assistance And Navigation On The Campus Site													
Electric Vehicle Charging													
Car Parking													
Applications For Intelligent Services	X			X	X								
Data Mining For Personalized Experience And Services Recommendation		X	X	X	X				X		X	X	X
Surveillance Localization And Navigation Systems									X	X		X	
Real Time Space Use			X	X									
Understanding Perceptions On Smartness													
Monitoring People's Opinion													
Knowledge Transfer And Collaboration		X			X	X	X						
Ubiquitous Learning		X											
Gamification And Virtual Learning Environments													
Student Innovation													
Smart Classrooms And Conference Rooms													
Multifaceted Frameworks For Developing Smart Campuses	X		X						X				
Virtual Campuses For Planning And Simulation													
Decision Making And Management Systems										X			

Certifications And Benchmarking												
Open Data Practices							X	X				
Wireless Sensor Networks And Wireless Infrastructures			X	X	X	X	X	X				
Middleware And Iot Naming Scheme	X											
Context Awareness						X		X				
Data Storage Security And Attack Simulation	X					X						
Authentication												

Tabela 29 – Relação de estudos aprendizado federado com categorias de estudos em *smart campus* - Parte 2 - Estudos 13 a 23

SMART CAMPUS CATEGORIES	**SFMEI**	**STUDIES**									
		13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Analyses Of Current Building Stock And Building Refurbishment												
Big Data Platforms And Frameworks					X		X					
Integration Of Real Time Data Collection Into Bim Environment				X								
Artificial Intelligence For Pattern Recognition Fault Detection And Control	X			X		X		X	X	X	X	X
Role Of Occupants				X					X			
Microgrids												
Campus Scale Sustainability Initiatives	X	X		X								
Waste Management And Recycling	X											
Campus Environmental Monitoring	X											
Monitoring And Understanding Mobility On Campuses		X										
Bus Transportation												
Assistance And Navigation On The Campus Site												
Electric Vehicle Charging												
Car Parking												
Applications For Intelligent Services	X	X										
Data Mining For Personalized Experience And Services Recommendation			X	X								
Surveillance Localization And Navigation Systems		X										
Real Time Space Use		X										
Understanding Perceptions On Smartness												
Monitoring People's Opinion												
Knowledge Transfer And Collaboration			X	X	X	X	X	X				
Ubiquitous Learning			X									
Gamification And Virtual Learning Environments												
Student Innovation												
Smart Classrooms And Conference Rooms												

Multifaceted Frameworks For Developing Smart Campuses	X		X					X				
Virtual Campuses For Planning And Simulation												
Decision Making And Management Systems		X		X	X	X	X	X	X	X	X	X
Certifications And Benchmarking												
Open Data Practices												
Wireless Sensor Networks And Wireless Infrastructures					X	X	X			X	X	X
Middleware And Iot Naming Scheme	X											
Context Awareness					X	X						
Data Storage Security And Attack Simulation	X								X	X	X	X
Authentication												

APÊNDICE C – ARQUIVOS DE CONFIGURAÇÃO - *DOCKER-COMPOSE* E *DOCKERFILES*

Listing C.1 – docker-compose.yml

```
1  version: '3'
2  services:
3  # DATABASE
4
5    # db server #
6    middleware_mysql_server:
7      extends:
8        file: docker-compose-db.yaml
9        service: middleware_mysql_server
10
11   # db client #
12   middleware_mysql_client:
13     extends:
14       file: docker-compose-db.yaml
15       service: middleware_mysql_client
16
17 # RabbitMq
18   rabbitmq:
19     extends:
20       file: docker-compose-rabbitmq.yaml
21       service: rabbitmq
22
23 # WEB SERVICES
24 # SERVER #
25   app_server:
26     extends:
27       file: docker-compose-web-server.yaml
28       service: app_server
29
30 # CLIENTS #
31   app_client_1:
32     extends:
33       file: docker-compose-web-client.yaml
34       service: app_client_1
35
```

```
36   app_client_2:
37     extends:
38       file: docker-compose-web-client.yaml
39       service: app_client_2
40
41   app_client_3:
42     extends:
43       file: docker-compose-web-client.yaml
44       service: app_client_3
45
46   app_client_4:
47     extends:
48       file: docker-compose-web-client.yaml
49       service: app_client_4
50
51 networks:
52   customnetwork:
53     driver: bridge
54     ipam:
55       config:
56         - subnet: 172.20.0.0/16
```

Listing C.2 – docker-compose-db.yml

```
1  version: '3'
2  services:
3  # DATABASE
4    # db server
5    middleware_mysql_server:
6      platform: linux/amd64
7      image: mariadb:10.5.8
8      restart: always
9      container_name: db-server
10     ports:
11       - 8080:8080
12       - 3306:3306
13       - 33060:33060
14     environment:
15       - MYSQL_DATABASE=middleware-mysql-server
16       - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
17       - MYSQL_USER=${MYSQL_USER}
18       - MYSQL_PASSWORD=${MYSQL_PASSWORD}
```

```
19        - MYSQL_ALLOW_EMPTY_PASSWORD=${MYSQL_ALLOW_EMPTY_PASSWORD}
20      volumes:
21        - ./datasets/categories:/datasets/categories:ro
22        - ./datasets/:/var/lib/mysql-files/
23        - ./server/db-server/mysql.cnf:/etc/mysql/my.cnf
24        - ./server/db-server/init:/docker-entrypoint-initdb.d
25      command: --performance-schema --local-infile
            --default-authentication-plugin=mysql_native_password
26      networks:
27        customnetwork:
28          ipv4_address: 172.20.10.1
29      deploy:
30        resources:
31          limits:
32            memory: 4000M
33          reservations:
34            memory: 200M
35
36    # db client
37    middleware_mysql_client:
38      platform: linux/amd64
39      image: mariadb:10.5.8
40      restart: always
41      container_name: db-client
42      ports:
43        - 8081:8080
44        - 3307:3306
45        - 33061:33060
46      environment:
47        - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
48        - MYSQL_DATABASE=${MYSQL_DATABASE}
49        - MYSQL_USER=${MYSQL_USER}
50        - MYSQL_PASSWORD=${MYSQL_PASSWORD}
51        - MYSQL_ALLOW_EMPTY_PASSWORD=${MYSQL_ALLOW_EMPTY_PASSWORD}
52      volumes:
53        - ./datasets/categories:/datasets/categories:ro
54        - ./datasets/:/var/lib/mysql-files/
55        - ./client/db-client/mysql.cnf:/etc/mysql/my.cnf
56        - ./client/db-client/init:/docker-entrypoint-initdb.d
57      command: --performance-schema --local-infile
            --default-authentication-plugin=mysql_native_password
58      networks:
```

```
59      customnetwork:
60        ipv4_address: 172.20.10.2
61
62  networks:
63    customnetwork:
64      driver: bridge
65      ipam:
66        config:
67          - subnet: 172.20.0.0/16
```

Listing C.3 – docker-compose-rabbitmq.yml

```
1   version: "2"
2   services:
3     rabbitmq:
4       image: rabbitmq:3-management
5       container_name: 'rabbitmq'
6       environment:
7         RABBITMQ_DEFAULT_VHOST: rabbitmq-host
8       expose:
9         - 15672
10      ports:
11          - 5672:5672
12          - 15672:15672
13      networks:
14        customnetwork:
15          ipv4_address: 172.20.255.0
16
17  networks:
18    customnetwork:
19      driver: bridge
20      ipam:
21        config:
22          - subnet: 172.20.0.0/16
23          # subnet range 172.20.0.1 - 172.20.255.254
```

Listing C.4 – docker-compose-web-server.yml

```
1   version: '3'
2   services:
3   # WEB SERVICES
4     app_server:
```

```
5
6     image: app-server-flask
7     restart: always
8     container_name: app-server
9     build:
10      context: .
11      dockerfile: Dockerfile.webserver
12    ports:
13      - 5678:5678
14      - 5001:5000
15    environment:
16      - CATEGORIES=co2,energy_consumption,temperature,water
17      - DB_SERVER_ADDRESS=172.20.10.1
18      - DB_SERVER_PORT=3306
19      - DB_SERVER_USER=test
20      - DB_SERVER_PASSWORD=test
21      - DB_SERVER_SCHEMA=middleware-mysql-server
22      - BROKER_ADDRESS=172.20.255.0
23      - BROKER_PORT=5678
24      - RUN_SCHEDULE_SERVER=${RUN_SCHEDULE_SERVER}
25      - JOB_STARTUP_INTERVAL_SERVER=${JOB_STARTUP_INTERVAL_SERVER}
26    logging:
27      driver: gelf
28      options:
29        gelf-address: "udp://localhost:12201" # Logstash UDP input port
30        tag: "app_server"
31    volumes:
32      - ./datasets/categories:/datasets/categories:ro
33    networks:
34      customnetwork:
35        ipv4_address: 172.20.20.1
36
37 networks:
38   customnetwork:
39     driver: bridge
40     ipam:
41       config:
42         - subnet: 172.20.0.0/16
```

Listing C.5 – docker-compose-web-client.yml

```
1 version: '3'
```

```
services:
####### 1 #######
  app_client_1:

    image: app-client-flask
    build:
      context: .
      dockerfile: Dockerfile.webclient
    restart: always
    container_name: app-client-1
    ports:
      - 5679:5678
      - 5002:5000
      - 81:80
    expose:
      - 5678
      - 5000
      - 80
    environment:
      - CLIENT_ID=1
      - DB_CLIENT_ADDRESS=172.20.10.2
      - CATEGORIES=${CATEGORIES}
      - DB_CLIENT_PORT=${DB_CLIENT_PORT}
      - DB_CLIENT_USER=${DB_CLIENT_USER}
      - DB_CLIENT_PASSWORD=${DB_CLIENT_PASSWORD}
      - DB_CLIENT_SCHEMA=${DB_CLIENT_SCHEMA}1
      - MDW_ADDRESS=${MDW_ADDRESS}
      - MDW_PORT=${MDW_PORT}
      - RUN_SCHEDULE=${RUN_SCHEDULE}
      - RUN_MQTT=${RUN_MQTT}
      - DOCKER_TLS_VERIFY=
      - DOCKER_CERT_PATH=
    logging:
      driver: gelf
      options:
        gelf-address: "udp://localhost:12201" # Logstash UDP input port
        tag: "app_client_1"
    volumes:
      - ./datasets/categories:/datasets/categories:ro
      - /etc/ssl/certs/:/usr/local/share/ca-certificates/
      - /etc/ssl/certs/ca-certificates.crt:/etc/ssl/certs/ca-certificates.crt
    networks:
```

```
44      customnetwork:
45        ipv4_address: 172.20.20.2
46
47
48  ####### 2 #######
49    app_client_2:
50      image: app-client-flask
51      build:
52        context: .
53        dockerfile: Dockerfile.webclient
54      restart: always
55      container_name: app-client-2
56      ports:
57        - 5680:5678
58        - 5003:5000
59        - 82:80
60      expose:
61        - 5678
62        - 5000
63        - 80
64      environment:
65        - CLIENT_ID=2
66        - DB_CLIENT_ADDRESS=172.20.10.2
67        - CATEGORIES=${CATEGORIES}
68        - DB_CLIENT_PORT=${DB_CLIENT_PORT}
69        - DB_CLIENT_USER=${DB_CLIENT_USER}
70        - DB_CLIENT_PASSWORD=${DB_CLIENT_PASSWORD}
71        - DB_CLIENT_SCHEMA=${DB_CLIENT_SCHEMA}2
72        - MDW_ADDRESS=${MDW_ADDRESS}
73        - MDW_PORT=${MDW_PORT}
74        - RUN_SCHEDULE=${RUN_SCHEDULE}
75        - RUN_MQTT=${RUN_MQTT}
76        - DOCKER_TLS_VERIFY=
77        - DOCKER_CERT_PATH=
78      logging:
79        driver: gelf
80        options:
81          gelf-address: "udp://localhost:12201" # Logstash UDP input port
82          tag: "app_client_2"
83      volumes:
84        - ./datasets/categories:/datasets/categories:ro
85        - /etc/ssl/certs/:/usr/local/share/ca-certificates/
```

```
86        - /etc/ssl/certs/ca-certificates.crt:/etc/ssl/certs/ca-certificates.crt
87      networks:
88        customnetwork:
89          ipv4_address: 172.20.20.3
90
91
92  ####### 3 #######
93    app_client_3:
94      image: app-client-flask
95      build:
96        context: .
97        dockerfile: Dockerfile.webclient
98      restart: always
99      container_name: app-client-3
100     ports:
101       - 5681:5678
102       - 5004:5000
103       - 83:80
104     expose:
105       - 5678
106       - 5000
107       - 80
108     environment:
109       - CLIENT_ID=3
110       - DB_CLIENT_ADDRESS=172.20.10.2
111       - CATEGORIES=${CATEGORIES}
112       - DB_CLIENT_PORT=${DB_CLIENT_PORT}
113       - DB_CLIENT_USER=${DB_CLIENT_USER}
114       - DB_CLIENT_PASSWORD=${DB_CLIENT_PASSWORD}
115       - DB_CLIENT_SCHEMA=${DB_CLIENT_SCHEMA}3
116       - MDW_ADDRESS=${MDW_ADDRESS}
117       - MDW_PORT=${MDW_PORT}
118       - RUN_SCHEDULE=${RUN_SCHEDULE}
119       - RUN_MQTT=${RUN_MQTT}
120       - DOCKER_TLS_VERIFY=
121       - DOCKER_CERT_PATH=
122     logging:
123       driver: gelf
124       options:
125         gelf-address: "udp://localhost:12201" # Logstash UDP input port
126         tag: "app_client_3"
127     volumes:
```

```
128       - ./datasets/categories:/datasets/categories:ro
129       - /etc/ssl/certs/:/usr/local/share/ca-certificates/
130       - /etc/ssl/certs/ca-certificates.crt:/etc/ssl/certs/ca-certificates.crt
131     networks:
132       customnetwork:
133         ipv4_address: 172.20.20.4
134
135
136 ####### 4 #######
137   app_client_4:
138     image: app-client-flask
139     build:
140       context: .
141       dockerfile: Dockerfile.webclient
142     restart: always
143     container_name: app-client-4
144     ports:
145       - 5682:5678
146       - 5005:5000
147       - 84:80
148     expose:
149       - 5678
150       - 5000
151       - 80
152     environment:
153       - CLIENT_ID=4
154       - DB_CLIENT_ADDRESS=172.20.10.2
155       - CATEGORIES=${CATEGORIES}
156       - DB_CLIENT_PORT=${DB_CLIENT_PORT}
157       - DB_CLIENT_USER=${DB_CLIENT_USER}
158       - DB_CLIENT_PASSWORD=${DB_CLIENT_PASSWORD}
159       - DB_CLIENT_SCHEMA=${DB_CLIENT_SCHEMA}4
160       - MDW_ADDRESS=${MDW_ADDRESS}
161       - MDW_PORT=${MDW_PORT}
162       - RUN_SCHEDULE=${RUN_SCHEDULE}
163       - RUN_MQTT=${RUN_MQTT}
164       - DOCKER_TLS_VERIFY=
165       - DOCKER_CERT_PATH=
166     logging:
167       driver: gelf
168       options:
169         gelf-address: "udp://localhost:12201" # Logstash UDP input port
```

```
170         tag: "app_client_4"
171     volumes:
172       - ./datasets/categories:/datasets/categories:ro
173       - /etc/ssl/certs/:/usr/local/share/ca-certificates/
174       - /etc/ssl/certs/ca-certificates.crt:/etc/ssl/certs/ca-certificates.crt
175     networks:
176       customnetwork:
177         ipv4_address: 172.20.20.5
178
179 networks:
180   customnetwork:
181     driver: bridge
182     ipam:
183       config:
184         - subnet: 172.20.0.0/16
```

Listing C.6 – Dockerfile.webserver

```
1  # Base image
2  FROM python:3.9-slim
3
4
5  # Set working directory
6  WORKDIR /app
7
8  # Copy the requirements file to the working directory
9  COPY requirements.txt .
10
11 # Install dependencies
12 RUN pip install -r requirements.txt
13
14 # Copy the application code to the working directory
15 COPY ./server .
16
17 # Set environment variables, if needed
18 ENV FLASK_APP=./server/app.py
19 ENV FLASK_RUN_HOST=0.0.0.0
20 # ENV FLASK_DEBUG=1
21
22 # Expose the port that the Flask application will run on
23 EXPOSE 5000
24 EXPOSE 5678
```

```
25 EXPOSE 80
26
27 # Start the Flask application and run debugpy on app
28 # CMD python3 -u -m debugpy --listen 0.0.0.0:5678 -m flask run --host=0.0.0.0
       --port 5000
29 # CMD python3 -u -m debugpy --wait-for-client --listen 0.0.0.0:5678 -m flask
       run --host=0.0.0.0 --port 5000
30
31 # CMD python3 -u -m debugpy --wait-for-client --listen 0.0.0.0:5678 -m flask
       run --port 5000
32 CMD python3 -u -m debugpy --listen 0.0.0.0:5678 -m flask run --port 5000
```

Listing C.7 – Dockerfile.webclient

```
1  # Base image
2  FROM python:3.9-slim
3
4  # Set working directory
5  WORKDIR /app
6
7  # Copy the requirements file to the working directory
8  COPY requirements.txt .
9
10 # Install dependencies
11 RUN pip install -r requirements.txt
12
13 # Copy the application code to the working directory
14 COPY ./client .
15
16 # Set environment variables, if needed
17 ENV FLASK_APP=./client/app.py
18 ENV FLASK_RUN_HOST=0.0.0.0
19 ENV FLASK_DEBUG=1
20
21 # Expose the port that the Flask application will run on
22 EXPOSE 5000
23 EXPOSE 5678
24 EXPOSE 80
25
26 # Start the Flask application and run debugpy on app
27 #
28 # CMD python3 -u -m debugpy --wait-for-client --listen 0.0.0.0:5678 -m flask
```

```
    run --host=0.0.0.0 --port 5000
29 CMD python3 -u -m debugpy --listen 0.0.0.0:5678 -m flask run --host=0.0.0.0
    --port 5000
30 # CMD python3 -m debugpy --listen 0.0.0.0:5678 -m flask run --host=0.0.0.1
    --port 9988
31 # CMD python3 -m debugpy --listen 0.0.0.0:5679 -m flask run --port 5001
```

APÊNDICE D – DETALHEMENT ENDPOINTS *SFMEI*

Nos detalhes desta seção os exemplos estão representados de maneira que acima da marcação "– – –" encontra-se a requisição e abaixo a resposta, como exemplificado a seguir, de uma requisição para um recurso que retorna o endereço de uma imagem aleatória de um cachorro:

```
GET https://random.dog/woof.json
Accept: application/json
Body:
 ---
200 OK
Content-Type: application/json
Response:
{
    "fileSizeBytes":117035,
    "url":"https://random.dog/2664e66d-eb92-434f-aab0-2e8a267032bf.jpg"
}
```

Os *endpoint* locais do *SFMEI Local Client* são:

- **/model/train/run** - Adiciona a configuração de um modelo candidato a treinamento, caso não exista. Ou, caso exista, altera a configuração deste modelo para um modelo apto a treinamento. Envia no corpo da requisição os parâmetros: (i) *model_name* - Nome do modelo a ser utilizado como referência. (ii) *category_name* - Nome da categoria do modelo. (iii) *context_table_name*- Nome da tabela de contexto que armazena os dados utilizados para treinamento do modelo.

```
POST https://localhost:8080/model/train/run
Accept: application/json
Body:
{
    "model_name": "nome_do_modelo",
    "category_name": "nome_da_categoria",
    "context_table_name: "nome_tabela_de_contexto"
}
---
200 OK
```

```
Content-Type: application/json
Response:
{}
```

- **/model/train/stop** - Altera a configuração do modelo para um modelo não apto a treinamento. Isso resulta que o modelo não será considerado para treinamento em nenhuma rodada de aprendizagem federada. Envia no corpo da requisição os parâmetros: (*i*) *model_name* - Nome do modelo a ser utilizado como referência. (*ii*) *category_name* - Nome da categoria do modelo. (*iii*) *context_table_name*- Nome da tabela de contexto que armazena os dados utilizados para treinamento do modelo.

```
POST https://localhost:8080/model/train/stop
Accept: application/json
Body:
{
    "model_name": "nome_do_modelo",
    "category_name": "nome_da_categoria",
    "context_table_name: "nome_tabela_de_contexto"
}
---
200 OK
Content-Type: application/json
Response:
{}
```

- **/model/update** - Adiciona novos dados manualmente na tabela de contexto. Envia no corpo da requisição os parâmetros: (*i*) *context_table_name*- Nome da tabela de contexto que armazena os dados utilizados para treinamento do modelo. (*ii*) *data* - Que possui um esquema de dados: (1) *instant* - Valor em timestamp do instante de leitura do dado. (2) *data* - Valor do dado lido.

```
POST https://localhost:8080/model/update
Accept: application/json
Body:
{
    "context_table_name": "nome_tabela_de_contexto",
    "data": [
        {
            "instant": "valor_timestamp_instante_leitura_do_dado",
```

```
            "data": "valor_do_dado"
        }
    ]
}
---
200 OK
Content-Type: application/json
Response:
{}
```

Os *endpoints* para consultas ao repositório do *SFMEI* são:

- **/categories** - Lista todos as categorias registradas no *SFMEI*. Na caixa abaixo, com a finalidade de exemplo, vemos a listagem de três categorias fictícias que são elas **nome_da_categoria_1** e **nome_da_categoria_2**.

```
GET https://localhost:8080/categories
Accept: application/json
---
200 OK
Content-Type: application/json
Response:
{
  [
    {
      "id": "id_categoria_1",
      "name": "nome_da_categoria_1",
      "detail": "detalhe_da_categoria_1"
    },
    {
      "id": "id_categoria_2",
      "name": "nome_da_categoria_2",
      "detail": "detalhe_da_categoria_2"
    }
  ]
}
```

- **/category** - Busca no repositório do *SFMEI* o detalhe de uma categoria, por nome, contendo a lista de todos os modelos treinados dessa categoria.

```
POST https://localhost:8080/category
Accept: application/json
Body:
{
    "category_name": "nome_da_categoria_1"
}
---
200 OK
Content-Type: application/json
Response:
{
  [
    {
      "id": "id_categoria_1",
      "name": "nome_da_categoria_1",
      "detail": "detalhe_da_categoria_1",
      "models" : [
        {
          "model_id": "id_do_modelo_1"
          "model_name": "nome_do_modelo_1",
          "score": "score_do_modelo_1",
          "category": "nome_da_categoria_1"
        },
        {
          "model_id": "id_do_modelo_2"
          "model_name": "nome_do_modelo_2",
          "score": "score_do_modelo_2",
          "category": "nome_da_categoria_2"
        }
      ]
    }
  ]
}
```

- **/model/get** - Busca um modelo pré-treinado por nome e categoria.

```
POST https://localhost:8080/model/get
Accept: application/json
Body:
```

```
{
    "model_name": "nome_do_modelo_1",
    "category_name": "nome_da_categoria_1"
}
---
200 OK
Content-Type: application/json
Response:
{
    "model_id": "id_do_modelo_1"
    "model_name": "nome_do_modelo_1",
    "score": "score_do_modelo_1",
    "category": "nome_da_categoria_1",
    "weights": "pesos_calculados_do_modelo_1",
    "process_date": "instante_do_treinamento_do_modelo_1"
}
```

Os *endpoints* a seguir relacionam-se de maneira direta com os *endpoints* para consultas no *SFMEI Local Client*:

- **/middleware/categories** - Lista todos as categorias registradas no SFMEI. Possui o mesmo ***Body*** e ***Response*** do endpoint **/categories** do *SFMEI Local Client*.

```
GET https://sfmei.example.address/middleware/categories
...
```

- **/middleware/category** - Busca no repositório do *SFMEI* o detalhe de uma categoria, por nome, contendo a lista de todos os modelos treinados dessa categoria. Possui o mesmo ***Body*** e ***Response*** do endpoint **/category** do *SFMEI Local Client*.

```
POST https://sfmei.example.address/middleware/category
...
```

- **/middleware/model/get** - Busca um modelo pré-treinado por nome e categoria. Possui o mesmo ***Body*** e ***Response*** do endpoint **/model/get** do *SFMEI Local Client*.

```
POST https://sfmei.example.address/middleware/get
...
```

Já os *endpoints* a seguir fazem parte de um conjunto de endereços de controle (registro e conexão) entre o *SFMEI Local Client* e o *SFMEI*:

- **/middleware/ready** - Verifica se o SFMEI está online e pronto para receber modelos para treinamento.

```
GET /middleware/ready
Accept: application/json
---
200 OK
Content-Type: application/json
Response:
{
    "ready": true
}
```

- **/middleware/register** - Registra clientes no middleware e retorna um identificador único de cliente e as configurações do *broker* de comunicação

```
PUT /middleware/register
Accept: application/json
---
200 OK
Content-Type: application/json
Response:
{
    "client_id": "id_unico_de_cliente",
    "broker": {
        "broker_address": "endereco_do_broker",
        "port": "porta_endereco_broker",
        "data_topic": "nome_do_topico_de_dados",
        "events_topic": "nome_do_topico_de_eventos"
    }
}
```

- **/middleware/connect** - Conecta um cliente registrado ao *SFMEI*.

```
POST /middleware/connect
Accept: application/json
Body:
```

```
{
    "client_id": "id_unico_de_cliente"
}
---
200 OK
Content-Type: application/json
Response:
{
    "message": "connected"
}
```

APÊNDICE E – SCRIPTS TRATAMENTO CONJUNTOS DE DADOS

In [ ]:
```
! pip install -q pandas
! pip install -q numpy
```

In [ ]:
```
import pandas as pd
import numpy as np
import random

import matplotlib.pyplot as plt

from sklearn import preprocessing

from collections import Counter
```

In [ ]:
```
df = pd.read_csv('./oulu-smartcampus-release-2021062801/application.csv',
                 usecols=['deveui','time','co2'],
                 low_memory=True)
print(df.index.size)
df = df[['deveui','time','co2']]
df.head()
```

In [ ]:
```
Deveui = df["deveui"]
Devices = dict(Counter(Deveui))
len(Devices)
```

In [ ]:
```
df = df[df['co2'].notna()]
qty_of_people = [1,2,3,4,5,6,7,8,9,10,12,20]
print(df.index.size)
df.head()
```

In [ ]:
```
df['time'] = df['time'].apply(str).str[:-3]
```

In [ ]:
```
device_ids = df['deveui'].unique()
qty = 5
p = int(len(device_ids) / qty) + 1
p
```

In [ ]:
```
ids = random.choices(df['deveui'].unique(), k=15)
```

In [ ]:
```
df.columns = ['dataid', 'instant', 'data']
df.head()
```

In [ ]:
```
ids[0]
```

In [ ]:
```
ids
```

In [ ]:
```
for index, id_ in enumerate(ids):
    df_1 = df[df['dataid'] == ids[index]].reset_index()[['instant','data']]
    df_1.to_csv("categories/co2/new_2_dataset_client_"+str(index)+".csv", index=False)
```

In [ ]:
```
def plotBeforeNormalization(df):
    df.plot(figsize=(16,4),legend=True)
    plt.title('data - BEFORE NORMALIZATION')
    plt.show()
```

In [ ]:
```
df = pd.read_csv(r'categories/co2/dataset_client_0.csv')
df.head()
```

In [ ]:
```
df.index = df['instant']
df = df[['data']]


plotBeforeNormalization(df)
```

In [ ]:
```
scaler = sklearn.preprocessing.MinMaxScaler()
df['data']=scaler.fit_transform(df['data'].values.reshape(-1,1))

df.head()
```

In [ ]:
```
plotBeforeNormalization(df)
```

In [ ]:
```
x_train = []
y_train = []
for i in range(20, len(df)):
    x_train.append(df.iloc[i-20 : i, 0])
    y_train.append(df.iloc[i, 0])
```

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
df = pd.read_csv('./15minute_data_austin/15minute_data_austin.csv',
                 usecols=['dataid','local_15min','grid'], low_memory=True)
df.head()
```

In [ ]:
```
df_ = df
```

In [ ]:
```
df_ = df_[df_['grid'].notna()]
df_['date'] = None
df_['aux'] = "00:00"
df_['date'] = df_['local_15min'].str.extract(r"^(2018-.[0-9]-.[0-9] .[0-9]:)")[0].str.ca
df_['instant'] = pd.to_datetime(df_['date']).values.astype(np.int64) // 10 ** 9
df_grouped = df_.groupby(['dataid', 'instant'], as_index=False)["grid"].sum()
df_grouped.columns = ['dataid', 'instant', 'data']
```

In [ ]:
```
ids = [9922,7901,8156,8386,8565]
```

In [ ]:
```
for x, y in enumerate(ids):
    print(x, y)
```

In [ ]:
```
for x, y in enumerate(ids):
    df_1 = df_grouped[df_grouped['dataid'] == y].reset_index()[['instant','data']]
    df_1.to_csv("categories/energy_consumption_a/dataset_client_"+str(x)+".csv", index=F
```

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
df = pd.read_csv('odd/environment.csv')
df['aux'] = "00:00"
df['date_aux'] = df['read_dt'].str.cat(df['read_tm'], sep=" ")
df['date'] = df['date_aux'].str.extract(r"^(.*[0-9]-.[0-9]-.[0-9] .[0-9]:)")[0].str.cat(
df['instant'] = pd.to_datetime(df['date']).values.astype(np.int64) // 10 ** 9
df.head()
```

In [ ]:
```
df_ = df[['equipment_id','instant','temperature']]
df_.columns = ['dataid', 'instant', 'data']
df_
```

In [ ]:
```
df_grouped = df_.groupby(['dataid', 'instant'], as_index=False)["data"].mean()
df_grouped.head()
```

In [ ]:
```
df_grouped['dataid'].unique()
```

In [ ]:
```
ids = ['sen-hwy','sen-dnr','sen-sbr','sen-lvr','sen-grg']
```

In [ ]:
```
for x, y in enumerate(ids):
    print(x, y)
```

In [ ]:
```
for x, y in enumerate(ids):
    df_1 = df_grouped[df_grouped['dataid'] == y].reset_index()[['instant','data']]
    df_1.to_csv("categories/temperature/dataset_client_"+str(x)+".csv", index=False)
```

In [1]:
```
files = ['Water_DWW.csv',
         'Water_HTW.csv',
         'Water_WHW.csv']
path = 'AMPDS2/'
```

In [2]:
```
import pandas as pd
```

In [3]:
```
for index, file in enumerate(files):
    aux = pd.read_csv('AMPDS2/'+file)
    aux = aux[['unix_ts','counter']]
    aux.columns = ['instant','data']
    aux = aux[aux['data'] != 0]
    aux.to_csv("categories/water/dataset_client_"+str(index)+".csv", index=False)
```

APÊNDICE F – SCRIPTS SIMULAÇÃO PREDIÇÃO

Análise Predição

In [1]:
```
pip install -q tensorflow
```

```
Note: you may need to restart the kernel to use updated packages.
```

Importação de Bibliotecas

In [2]:
```
import gc
import json
import requests
import tensorflow as tf

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from keras import backend as k
from keras.callbacks import Callback

import sklearn.preprocessing

import time

from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
from sklearn.metrics import r2_score
```

Classe limpar garbage collector

In [3]:
```
class ClearMemory(Callback):
    def on_epoch_end(self, epoch, logs=None):
        gc.collect()
        k.clear_session()
```

Requisição ao Middleware - SFMEI

In [4]:
```
url = "http://localhost:5002/model/get"

payload = json.dumps({
  "category": "co2",
  "model": "model_co2"
})
headers = {
  'Content-Type': 'application/json'
}
```

In [5]:
```
response = requests.request("POST", url, headers=headers, data=payload)
```

In [6]:
```
response.status_code
```

Out[6]:
```
200
```

In [ ]:
```
response.json()
```

Criar modelo

In [8]:
```
def create_model(model: Sequential(), shape):
    model.add(LSTM(units = 64, activation="tanh", return_sequences = True, input_shape =
    model.add(Dropout(0.2))
    model.add(LSTM(32, return_sequences=False))
    model.add(Dropout(0.2))

    model.add(Dense(1))
    return model
```

Pré-processamento de dados

In [9]:
```
def normalize(df):
    print('Normalizing data')
    scaler = sklearn.preprocessing.MinMaxScaler()
    df['data']=scaler.fit_transform(df['data'].values.reshape(-1,1))
    print('Data Normalized')
    return df
```

In [10]:
```
def load_data(stock, seq_len):
    stock_size = stock.index.size
    limit = stock_size = int(stock_size * 0.05)
    x_train = []
    y_train = []
    for i in range(seq_len, len(stock)):
        x_train.append(stock.iloc[i-seq_len : i, 0])
        y_train.append(stock.iloc[i, 0])

    #1 last days are going to be used in test
    x_test = x_train[limit:]
    y_test = y_train[limit:]

    #2 first 'limit' days are going to be used in training
    x_train = x_train[:limit]
    y_train = y_train[:limit]

    #3 convert to numpy array
    x_train = np.array(x_train)
    y_train = np.array(y_train)

    x_test = np.array(x_test)
    y_test = np.array(y_test)

    #4 reshape data to input into RNN models
    x_train = np.reshape(x_train, (limit, seq_len, 1))

    x_test = np.reshape(x_test, (x_test.shape[0], seq_len, 1))

    return [x_train, y_train, x_test, y_test]
```

Utils para transformar Json para Numpy Array

In [11]:
```
def parse_weights_from_json_to_array(weights_json):
    df_w = pd.read_json(weights_json, orient='index')
    weights = []
    for i in df_w.columns:
        arr = np.array(df_w[i].dropna().values.tolist())
        weights.append(arr)
    return weights
```

In [12]:
```
file_names = ["co2_dataset_client_X_untrained",
              "co2_dataset_client_Y_untrained",
              "co2_dataset_client_Z_untrained"]
```

Função de Predição

1. Função lê arquivo com conjunto de dados não treinados.
2. Em seguida normaliza dos dados, transforma em dados pra predição e teste.
3. Realiza predição baseado em dados de entrada
4. Avalia escore R2
5. Compara predição e conjunto de dados de teste.

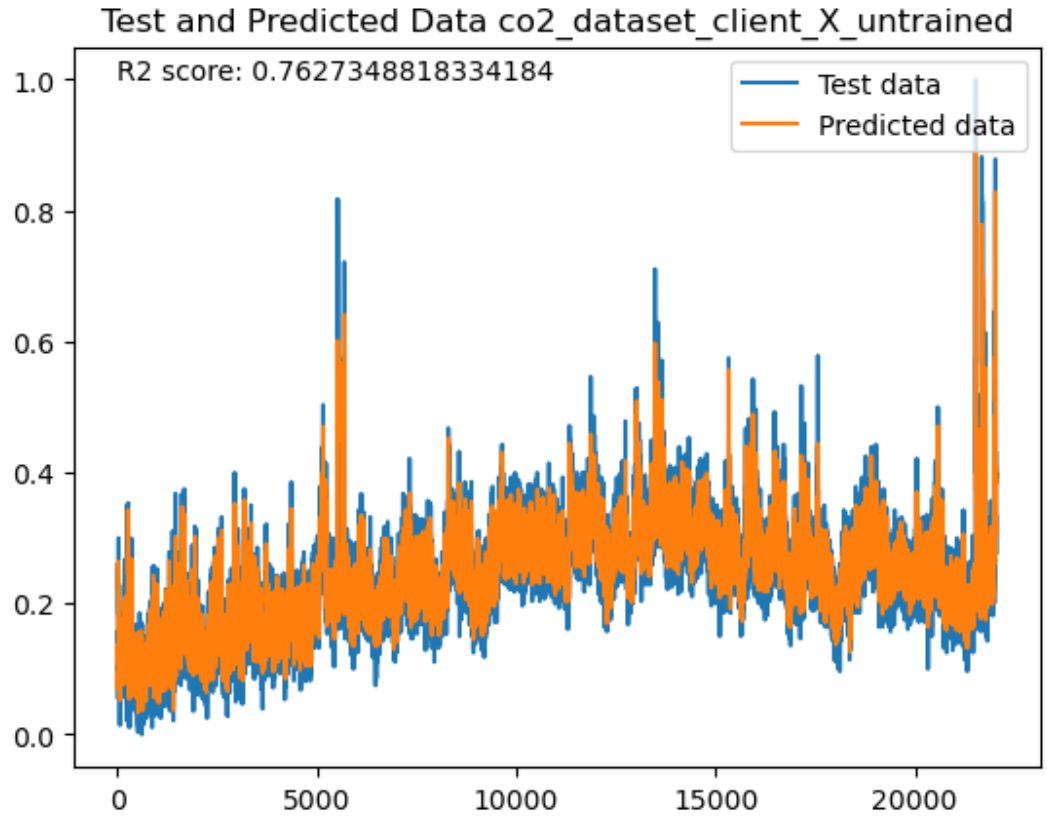
In [13]:
```
def predict(file_name):
    df_original = pd.read_csv(f'datasets_app_using/{file_name}.csv')
    df_original.index = df_original['instant']
    df_original = df_original[['data']]
    df_original.head()

    df = normalize(df_original)

    x_train, y_train, x_test, y_test = load_data(df, 20)

    model = create_model(Sequential(), x_train.shape[1])

    start = time.time()
    model.compile(optimizer="rmsprop", loss= tf.keras.losses.MeanSquaredError())
    finish = time.time()
    print(f'Finishing model compilation. Elapsed Time: {finish - start}')

    weights_json = response.json()['response']['weights']

    weights = parse_weights_from_json_to_array(weights_json)
    model.set_weights(weights)

    print('Starting model prediction')
    start = time.time()
    predictions = model.predict(x_test)
    score = r2_score(y_test, predictions)
    finish = time.time()
    print(f'Finishing model prediction. Elapsed Time: {finish - start}')

    plt.plot(y_test, label=f'Test data')
    plt.plot(predictions, label=f'Predicted data')
    plt.text(x=1, y=1, s=f'R2 score: {score}')
    plt.legend(loc='upper right')
    plt.title(f'Test and Predicted Data {file_name}')
    plt.savefig(f'datasets_app_using/{file_name}.pdf', bbox_inches='tight', format='pdf'
    plt.show()
```

Executa predição

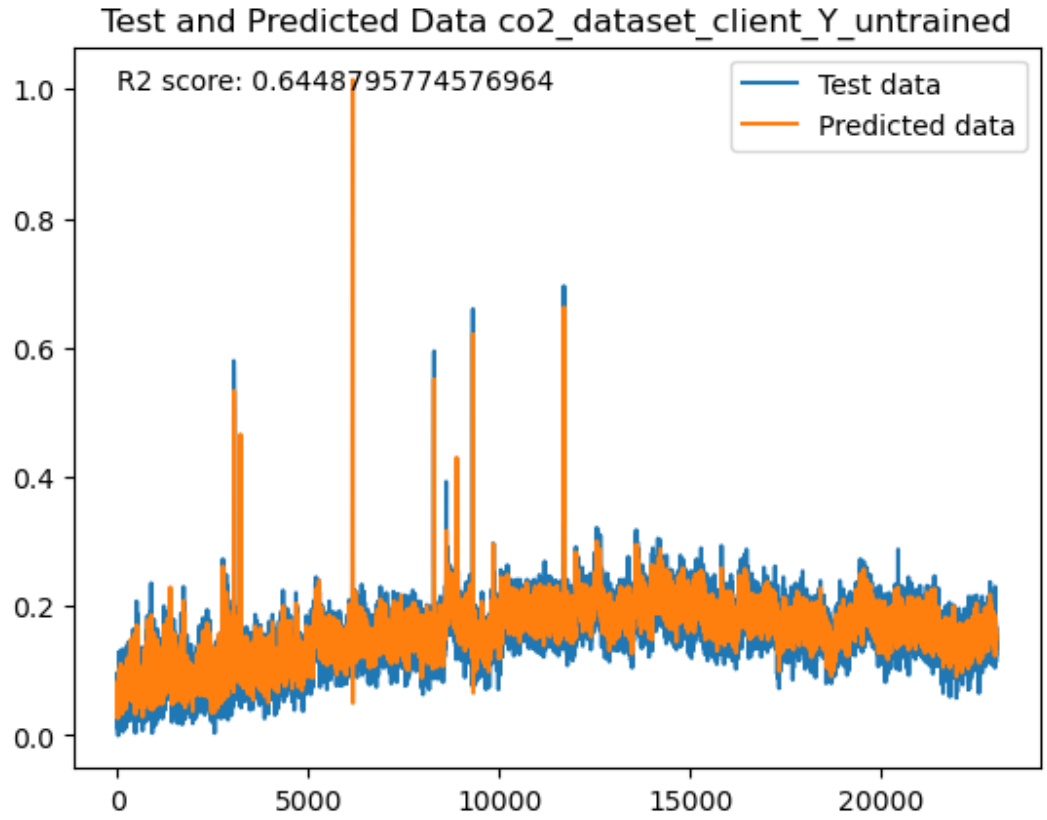
In [14]:
```
for file in file_names:
    predict(file)
```

```
Normalizing data
Data Normalized
/Users/ucasesantos/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rnn.py:20
4: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the model
```

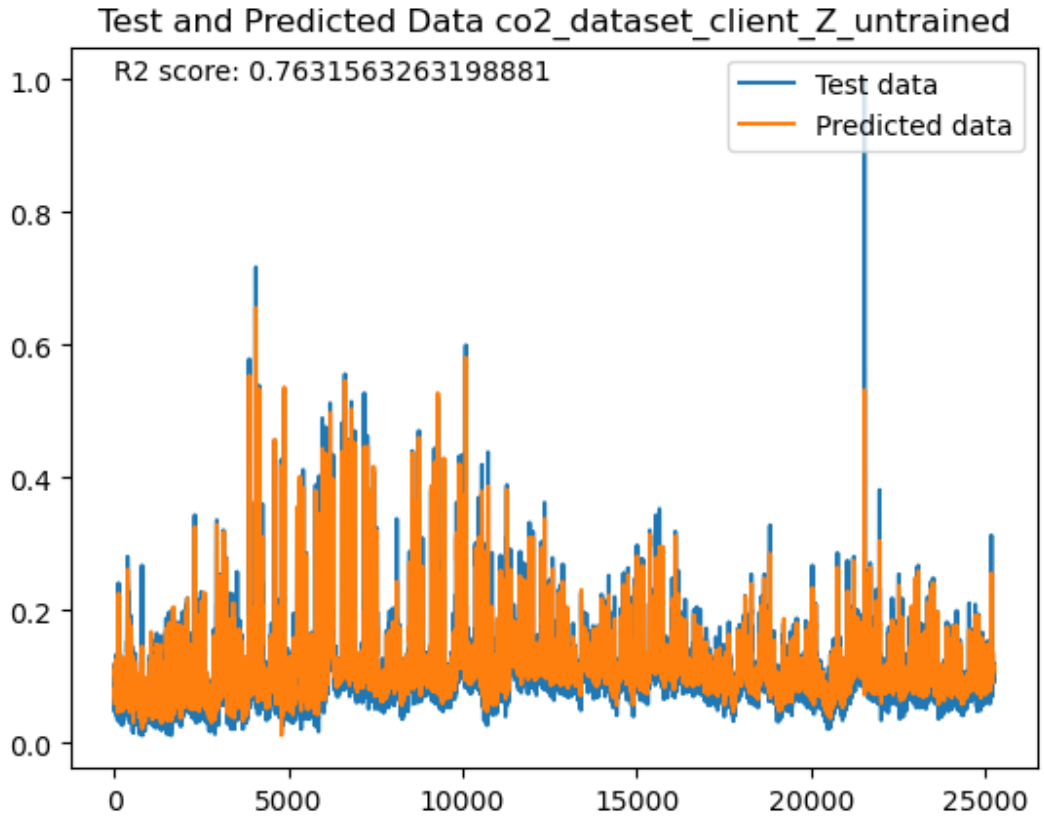
```
instead.
  super().__init__(**kwargs)
Finishing model compilation. Elapsed Time: 0.002458810806274414
Starting model prediction
689/689 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Finishing model prediction. Elapsed Time: 1.5608429908752441
```




```
Normalizing data
Data Normalized
Finishing model compilation. Elapsed Time: 0.0009388923645019531
Starting model prediction
  1/720 ━━━━━━━━━━━━━━━━━━━━ 1:30 125ms/step
/Users/ucasesantos/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
720/720 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
Finishing model prediction. Elapsed Time: 1.891767978668213
```




```
Normalizing data
Data Normalized
Finishing model compilation. Elapsed Time: 0.0010616779327392578
Starting model prediction
  1/789 ━━━━━━━━━━━━━━━━━━━━ 1:40 128ms/step
/Users/ucasesantos/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
789/789 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
Finishing model prediction. Elapsed Time: 1.8442919254302979
```


Test and Predicted Data co2_dataset_client_Z_untrained
R2 score: 0.7631563263198881
Test data
Predicted data
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000