



PEDRO GONÇALO DOS SANTOS VENTURA

Licenciado em Engenharia Eletrotécnica e de Computadores

Aprendizagem Federada em IoT: Capacitação de Dispositivos para Aquisição e Processamento de Dados

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
jan, 2024

Aprendizagem Federada em IoT: Capacitação de Dispositivos para Aquisição e Processamento de Dados

PEDRO GONÇALO DOS SANTOS VENTURA

Licenciado em Engenharia Eletrotécnica e de Computadores

Orientador: Ricardo Jardim Gonçalves,
Professor Catedrático, Universidade NOVA de Lisboa

Coorientador: Ruben Costa,
Professor Auxiliar Convidado, Universidade NOVA de Lisboa

Júri:

Presidente: Nome do(a) presidente,
Professor(a) Catedrático(a), FCT-NOVA

Arguentes: Pedro Pereira,
Professor Auxiliar, FCT-NOVA
Nome do outro arguente,
Professor Auxiliar, Ainda Outra Universidade

Orientador: Nome do orientador presente nas provas,
Professor Associado, FCT-NOVA

Membros: Mais um membro do júri,
Professora Catedrática, Outra Universidade
E ainda mais um membro do júri,
Professor Auxiliar, Ainda Outra Universidade

Aprendizagem Federada em IoT: Capacitação de Dispositivos para Aquisição e Processamento de Dados

Copyright © Pedro Ventura, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais,

Este documento foi criado com o processador de texto Microsoft Word e o template NOVATHESIS Word [11].

AGRADECIMENTOS

A realização desta dissertação marca o culminar de uma jornada académica que não teria sido possível sem o apoio e contribuição de várias pessoas às quais estou profundamente grato.

Em primeiro lugar, gostaria de expressar um agradecimento especial ao Professor Rúben Costa, pela sua orientação valiosa, conhecimentos partilhados e disponibilidade constante. A sua contribuição foi determinante para o desenvolvimento e conclusão desta dissertação. Expresso também a minha profunda gratidão ao Amin Khoramodari, cujo apoio e insights foram cruciais ao longo de todo este processo. A sua colaboração enriqueceu significativamente este trabalho.

Gostaria também de expressar a minha sincera gratidão à minha família, pelo seu apoio incondicional, encorajamento constante e compreensão ao longo de todo este percurso. Aos meus amigos, agradeço pela paciência, pelos momentos de descontração e pelo ânimo que me proporcionaram nos momentos mais desafiantes.

Aos investigadores do UNINOVA, o meu sincero agradecimento pela ajuda que me forneceram durante o desenvolvimento da dissertação. Os vossos conhecimentos e sugestões enriqueceram significativamente este trabalho.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização desta dissertação, o meu profundo agradecimento.

RESUMO

O *Federated Learning* (FL) surge como uma abordagem inovadora na área de *Machine Learning*, motivada pela necessidade crescente de processamento de dados mais próximo dos dispositivos. Esta técnica oferece várias vantagens teóricas, principalmente em termos de segurança e privacidade dos dados. A principal motivação para o uso do FL é a capacidade de treinar modelos de aprendizagem de máquina sem a necessidade de centralizar dados sensíveis. Isto é particularmente relevante em cenários onde a privacidade é crucial, como em ambientes de escritórios.

Nesta dissertação, serão abordados diversos aspetos do FL, incluindo os seus benefícios, desafios, bem como a sua taxonomia. O caso de estudo foca-se na previsão da qualidade do ambiente em escritórios, considerando fatores como temperatura, iluminância, qualidade do ar, humidade e ruído. A implementação do FL será desenvolvida utilizando dispositivos *edge*, e testada através de vários modelos de *Deep Learning* e regressão.

Os resultados demonstraram que modelos como xGBoost e Gradient Boosting apresentaram previsões consistentes para uma amostragem de 1 minuto, enquanto os modelos RNN e GRU foram eficazes para amostragens maiores, principalmente na de 10 minutos. Apesar de desafios como a descalibração do sensor da qualidade do ar e dados não estacionários, o FL revelou-se promissor para melhorar as condições de trabalho em edifícios de escritórios.

Implementações futuras podem considerar a integração do FL com sistemas de automação predial, como HVAC, permitindo ajustes em tempo real, expandindo assim a aplicação prática desta tecnologia.

Palavras chave: *Federated Learning, Machine Learning, IoT, edge devices.*

ABSTRACT

Federated Learning (FL) emerges as an innovative approach in the field of Machine Learning, driven by the growing need for data processing closer to the devices. This technique offers several theoretical advantages, primarily in terms of data security and privacy. The main motivation for using FL lies in its ability to train machine learning models without centralizing sensitive data. This is particularly relevant in scenarios where privacy is crucial, such as in office environments.

This dissertation addresses various aspects of FL, including its benefits, challenges, and taxonomy. The case study focuses on predicting office environmental quality, considering factors such as temperature, illuminance, air quality, humidity, and noise. The FL implementation is developed using edge devices and tested with various Deep Learning and regression models.

The results showed that models such as xGBoost and Gradient Boosting provided consistent predictions for a sampling rate of 1 minute, while RNN and GRU models were effective for larger sampling intervals, particularly at 10 minutes. Despite challenges such as air quality sensor miscalibration and non-stationary data, FL proved promising for improving working conditions in office buildings.

Future implementations may explore the integration of FL with building automation systems, such as HVAC, enabling real-time adjustments and thereby expanding the practical application of this technology.

Keywords: Federated Learning, Machine Learning, IoT, edge devices.

ÍNDICE

1	INTRODUÇÃO	1
1.1	Contexto e Motivação	1
1.2	Problema e objetivos	5
1.3	Abordagem e contribuições	6
1.4	Organização do documento	7
2	ESTADO DA ARTE	9
2.1	Princípios do FL	9
2.1.1	Definição e contextualização histórica	10
2.1.2	FL vs abordagens clássicas	13
2.1.3	Benefícios do FL	15
2.1.4	Desafios ligados ao FL	16
2.1.5	Aplicações de FL	18
2.2	Fundamentos Teóricos	20
2.2.1	Taxonomia do FL	21
2.2.2	Séries Temporais	29
2.2.3	FL em IoT	32
2.2.4	Algoritmos de agregação	36
2.2.5	Algoritmos	38

2.3	Trabalho Relacionado	40
3	SOLUÇÃO PROPOSTA	43
3.1	Dispositivo B-Health	43
3.1.1	Elementos do dispositivo B-Health.....	43
3.1.2	Configuração B-Health	46
3.2	Solução desenvolvida	49
3.2.1	Flower.....	50
4	ANÁLISE DE DADOS E DA QUALIDADE DO AMBIENTE	55
4.1	Análise de Dados	55
4.1.1	Disponibilidade.....	55
4.1.2	Análise gráfica e estatística dos dados.....	56
4.2	Análise da qualidade do ambiente	66
5	ANÁLISE DE RESULTADOS	69
5.1	Definição de parâmetros	69
5.2	Dados reais relativos ao horizonte de previsão	70
5.3	Modelos de referência	72
5.3.1	Facebook Prophet	72
5.3.2	Modelos de Regressão.....	74
5.4	Modelos de Deep Learning.....	78
5.4.1	Cenário 1 – Amostragem de 1 minuto.....	79
5.4.2	Cenário 1 – Amostragem de 5 minutos	82
5.4.3	Cenário 2 – Amostragem de 5 minutos	84
5.4.4	Cenário 1 – Amostragem de 10 minutos	86
5.4.5	Cenário 2 – Amostragem de 10 minutos	88

5.4.6	Outras Arquiteturas (Amostragens de 10 e 15 minutos)	90
6	CONCLUSÃO.....	101

ÍNDICE DE FIGURAS

Figura 1 - Market size do FL, por região entre 2018 e 2030 (Projeção de valores entre 2023 – 2030).....	3
Figura 2 - Market size do FL, por aplicação entre 2018 e 2030 (Projeção de valores entre 2023 – 2028).....	3
Figura 3 - Número de publicações que abordam o tema FL	5
Figura 4 - Metodologia	6
Figura 5 - Representação da linha temporal do ML	12
Figura 6 - Representação gráfica do HFL (adaptada de [30]).....	22
Figura 7 - Representação gráfica do VFL (adaptada de [30])	23
Figura 8 - Representação gráfica do FTL ou Hybrid FL (adaptada de [30])	24
Figura 9 - Topologia centralizada (adaptada de [30])	26
Figura 10 - Topologia descentralizada (adaptada de [30])	27
Figura 11 - Representação típica de um sistema FL-IoT (adaptada de [30])	34
Figura 12 - Visualização frontal da caixa B-Health.....	44
Figura 13 - Visualização lateral da caixa B-Health	45
Figura 14 - Diagrama de ligação da parte do cliente, desde os sensores até ao MQTT broker.....	47
Figura 15 - Interface de monitorização em tempo real de uma caixa B-Health.....	47
Figura 16 - Diagrama de ligação do lado do servidor, desde o MQTT Broker até à base de dados MariaDB.....	48
Figura 17 - Estrutura da solução implementada.....	50
Figura 18 – Dispositivo 1 - Iluminância: (a) Variação temporal; (b) Boxplot.....	59
Figura 19 - Dispositivo 2 - Iluminância: (a) Variação temporal; (b) Boxplot.....	59

Figura 20 - Dispositivo 3 - Iluminância: (a) Variação temporal; (b) Boxplot.....	59
Figura 21 – Dispositivo 1 - Ruído: (a) Variação temporal; (b) Boxplot	60
Figura 22 - Dispositivo 2 - Ruído: (a) Variação temporal; (b) Boxplot.....	60
Figura 23 - Dispositivo 3 - Ruído: (a) Variação temporal; (b) Boxplot.....	61
Figura 24 – Dispositivo 1 - Temperatura: (a) Variação temporal; (b) Boxplot.....	61
Figura 25 - Dispositivo 2 - Temperatura: (a) Variação temporal; (b) Boxplot.....	62
Figura 26 - Dispositivo 3 - Temperatura: (a) Variação temporal; (b) Boxplot.....	62
Figura 27 – Dispositivo 1 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot.....	63
Figura 28 - Dispositivo 2 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot.....	63
Figura 29 - Dispositivo 3 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot.....	63
Figura 30 – Dispositivo 1 - Humidade: (a) Variação Temporal; (b) Boxplot.....	64
Figura 31 - Dispositivo 2 - Humidade: (a) Variação Temporal; (b) Boxplot	64
Figura 32 - Dispositivo 3 - Humidade: (a) Variação Temporal; (b) Boxplot	64
Figura 33 – Dispositivo 1 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot.....	65
Figura 34 - Dispositivo 2 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot.....	65
Figura 35 - Dispositivo 3 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot.....	65
Figura 36 – Variação temporal da Qualidade do Ambiente (Avaliação Numérica) no horizonte de previsão para cada dispositivo: (a) Dispositivo 1; (b) Dispositivo 2; (c) Dispositivo 3	71
Figura 37 - Previsão do modelo Prophet referente a 6 horas para uma amostragem de 15 minutos	73
Figura 38 - Random Forest - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min.....	75
Figura 39 - ExtraTrees - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min.....	76
Figura 40 - xGBoost - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min.....	77
Figura 41 – Gradient Boosting - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min.....	77

Figura 42 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	81
Figura 43 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	81
Figura 44 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	83
Figura 45 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	84
Figura 46 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	85
Figura 47 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	86
Figura 48 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	87
Figura 49 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	88
Figura 50 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	89
Figura 51 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN	90
Figura 52 - Perda no treino e na validação do modelo RNN na Situação 1 para cada cenário: (a) Cenário 1; (b) Cenário 2	93
Figura 53 - Perda no treino e na validação do modelo RNN na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2	93
Figura 54 - Previsão dada pelo modelo RNN na Situação 1 em cada cenário: (a) Cenário 1; (b) Cenário 2.....	93
Figura 55 - Previsão dada pelo modelo RNN na Situação 3 em cada cenário: (a) Cenário 1; (b) Cenário 2.....	93
Figura 56 – Perda no treino e na validação do modelo LSTM: (a) Cenário 1 – Situação 1; (b) Cenário 2 – Situação 2.....	95
Figura 57 - Perda no treino e na validação do modelo LSTM na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2	95

Figura 58 - Previsão dada pelo modelo LSTM: (a) Cenário 1 – Situação 1; (b) Cenário 2 – Situação 2.....	96
Figura 59 - Previsão dada pelo modelo LSTM na Situação 3 em cada cenário: (a) Cenário 1; (b) Cenário 2.....	96
Figura 60 - Perda no treino e na validação do modelo GRU: (a) Cenário 1 – Situação 4; (b) Cenário 2 – Situação 1.....	98
Figura 61 - Perda no treino e validação do modelo GRU na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2.....	98
Figura 62 - Previsão dada pelo modelo GRU: (a) Cenário 1 – Situação 4; (b) Cenário 2 – Situação 1.....	98
Figura 63 - Previsão dada pelo modelo GRU na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2.....	99

ÍNDICE DE TABELAS

Tabela 1 - Associação entre sensores, variáveis e as suas unidades.....	45
Tabela 2 - Estrutura e tipo de dados de cada coluna referente a cada tabela dos sensores	48
Tabela 3 - Especificações dos Raspberry Pi's 3 e 4.....	49
Tabela 4 - Disponibilidade de cada sensor referente a cada caixa B-Health.....	56
Tabela 5 - Estatísticas dos dados referentes a cada sensor do dispositivo 1	57
Tabela 6 - Estatísticas dos dados referentes a cada sensor do dispositivo 2	57
Tabela 7 - Estatísticas dos dados referentes a cada sensor do dispositivo 3	57
Tabela 8 - Classificação para cada variável ambiental	66
Tabela 9 - Qualificação qualitativa e numérica da qualidade dos valores medidos pelos sensores.....	67
Tabela 10 - Intervalos numéricos da avaliação numérica tendo em conta a avaliação qualitativa	67
Tabela 11 - Tabela agrupada com um único intervalo de tempo, espaçado de 12 segundos, para cada sensor	68
Tabela 12 - Resultados obtidos pelo Facebook Prophet da função da perda para cada amostragem.....	73
Tabela 13 - Resultados obtidos para os modelos de regressão para cada amostragem	74
Tabela 14 - Resultados Cenário 1: Amostragem 1 minuto.....	80
Tabela 15 - Resultados Cenário 1: Amostragem 5 minutos	82
Tabela 16 - Resultados Cenário 2: Amostragem 5 minutos	84
Tabela 17 - Resultados Cenário 1: Amostragem 10 minutos.....	86
Tabela 18 - Resultados Cenário 2: Amostragem 10 minutos.....	88

Tabela 19 - Situações a simular com as diferentes arquiteturas para cada modelo de Deep Learning.....	91
Tabela 20 - Resultados RNN: Cenário 1	91
Tabela 21 - Resultados RNN: Cenário 2	92
Tabela 22 - Resultados LSTM: Cenário 1.....	94
Tabela 23 - Resultados LSTM: Cenário 2.....	94
Tabela 24 - Resultados GRU: Cenário 1.....	96
Tabela 25 - Resultados GRU: Cenário 2.....	97

GLOSSÁRIO

Cliente	Qualquer dispositivo que esteja integrado na rede.
Dispositivo	Componente de hardware físico ou equipamento que fornece uma ou mais funções de computação dentro de um sistema.
Edge device	Dispositivo que realiza, total ou parcialmente, o processamento de dados localmente.
Embedded device	Dispositivo independente responsável por executar uma tarefa específica, dentro de um sistema computacional mais abrangente.
Epochs	Número total de iterações de todos os dados de treino num ciclo para treinar o modelo de ML.
Framework	Estrutura ou arquitetura base que fornece funcionalidades fundamentais e design pré-definido, de forma a facilitar o desenvolvimento do sistema.
Micro-moments	Instâncias ou casos de quando o consumidor necessita espontaneamente de um dispositivo, para responder a uma certa necessidade.
Overflow	Corrupção da memória devido à escrita de dados em posições de memória que já estão ocupadas.
Overfitting	Dificuldade na generalização do modelo, mantendo-se demasiado próximo dos dados de treino.
Pretraining	Processo de treino de um modelo em vastos conjuntos de dados, permitindo a este aprender vários padrões e características relevantes em vários

domínios. Este novo conhecimento pode ser depois aplicado e adaptado para algumas tarefas em específico.

Stragglers

Nó ou dispositivo significativamente mais lento que os restantes, o que causa atrasos na computação geral.

SIGLAS

ADAGRAD	Adaptive Gradient Algorithm.
ADAM	Adaptive Moment Estimation.
AIoT	Artificial Internet of Things.
CIFAR-10	Canadian Institute For Advanced Research.
DL	Deep Learning
DNN	Deep Neural Network.
(EM^3)	Consumer Engagement towards Energy Saving Behaviour by Means of Exploiting Micro Moments and Mobile Recommendation Systems.
FedAvg	Federated Averaging.
FedGRU	Federated Gated Recurrent Unit.
FedUA	Federated User Authentication.
FL	Federated Learning.
FLS	Federated Learning System.
FTL	Federated Transfer Learning.
GAN	Generative Adversarial Networks.
GBDT	Gradient Boosting Decision Trees.
GRU	Gated Recurrent Unit.
HVAC	Heating, Ventilating and Air Conditioning.

HFL	Horizontal Federated Learning.
IA	Inteligência Artificial.
IEEE	Institute of Electrical and Electronics Engineers.
IoT	Internet of Things.
LASSO	Least Absolute Shrinkage and Selection Operator.
LSTM	Long Short-Term Memory.
ML	Machine Learning.
MNIST	Modified National Institute of Standards and Technology.
NN	Neural Network.
P2P	Peer-to-Peer.
RGPD	Regime Geral sobre a Proteção de Dados.
RNN	Recurrent Neural Network.
SAE	Sparse Autoencoder.
SMC	Secure Multiparty Computation.
SimFL	Similarity-based Federated Learning.
SGD	Stochastic Gradient Descent.
SNARC	Stochastic Neural Analog Reinforcement Calculator.
SVM	Support Vector Machine.
TEE	Trusted Execution Environment.
VFL	Vertical Federated Learning.

INTRODUÇÃO

Neste capítulo, serão abordados o contexto e a motivação subjacentes a esta dissertação. Serão apresentados os desafios, que não servirão apenas de impulsionador para a formulação desta dissertação, mas também serão delineadas uma pergunta de investigação e uma hipótese como resposta. Posteriormente será discutida a abordagem a seguir, de forma a orientar a análise e pesquisa. Por fim, será apresentada a organização do documento.

1.1 Contexto e Motivação

A era digital trouxe consigo oportunidades notáveis e desafios formidáveis. No epicentro desta era encontra-se o ML (*Machine Learning*), um impulsionador por trás dos avanços tecnológicos que têm a capacidade de reconfigurar a indústria e aprimorar as nossas vidas. A capacidade do ML processar vastas quantidades de dados, identificar padrões ocultos e impulsionar a tomada de decisões é evidente em diversos domínios, desde a saúde, finanças, entre outros. É o motor que impulsiona o nosso mundo orientado por dados, com o Departamento de Estatísticas do Trabalho dos EUA a prever um substancial crescimento de 23% no emprego nesta área entre 2022 e 2032 [1].

Enquanto a influência do ML cresce, surgem não apenas preocupações sobre a privacidade dos dados, mas também a necessidade de otimizar a comunicação ao reduzir sobrecargas na mesma, bem como promover a sua eficiência energética.

A privacidade dos dados não é apenas uma questão ética, é também uma questão legal. A conformidade com regulamentos de proteção de dados como o RGPD [2] (Regime Geral sobre a Proteção de Dados) é tanto uma exigência como uma salvaguarda contra multas e consequências legais. Mitigar violações de dados, prevenir a exploração de dados, proteger os direitos humanos e combater a discriminação são todos componentes integrais da preservação

da privacidade dos dados. Além disso, a segurança de dados está intimamente ligada à privacidade dos dados, uma vez que ambas se complementam para proteger os dados contra acessos não autorizados e ameaças cibernéticas [3]. Embora a privacidade dos dados seja uma questão central a ser abordada, a abordagem a considerar deve ter em conta também a otimização do fluxo da comunicação bem como a sua eficiência, entre outros fatores.

De forma a desenvolver essas abordagens, o conceito de descentralização de dados surge como uma necessidade crítica. A descentralização de dados, inicialmente explorada como uma resposta à centralização focada no armazenamento de dados em grandes repositórios, surge como uma solução não apenas para proteger a privacidade, mas também para reduzir a sobrecarga na comunicação e promover a sua eficiência. Ao distribuir dados por diversos locais e dispositivos, a descentralização não apenas reduz os riscos associados a falhas únicas, mas também se alinha com princípios de comunicação eficazes.

No contexto de ML, a descentralização de dados não apenas melhora a proteção da privacidade, mas também otimiza o fluxo de comunicação entre dispositivos [4]. Em vez de agregar grandes conjuntos de dados num local central, os modelos de ML podem ser treinados em fontes de dados descentralizadas. Isto reduz a necessidade de partilhar amplamente dados sensíveis, diminuindo o risco de violações de privacidade e aumentando a proteção das informações pessoais, minimizando a sobrecarga de comunicação dos dispositivos.

Embora as implementações descentralizadas ofereçam vantagens, como a distribuição de dados em diversos locais, também enfrentam desafios, como a interoperabilidade, bem como a necessidade da existência de uma rede de comunicações eficiente. Esta refere-se à capacidade de um sistema interagir e comunicar com outro. Os desafios incluem a heterogeneidade de dados e dispositivos, questões de segurança e privacidade, problemas de compatibilidade e a necessidade de coordenação nas atualizações. Todos estes desafios serão abordados mais à frente. Notar que, embora estes sejam desafios relacionados ao tema desta dissertação, a sua resolução está fora do âmbito deste trabalho.

Em conclusão, à medida que o ML continua a moldar o mundo orientado por dados, a crescente importância da privacidade e descentralização dos mesmos estão entrelaçadas, bem como a otimização da comunicação e eficiência energética. Encontrar um equilíbrio entre o potencial de ML e a proteção da informação pessoal é crucial, alinhando-se com o cenário em constante evolução da tecnologia e dados.

No âmbito destes desafios, o *Federated Learning* (FL) destaca-se como uma solução promissora, concentrando-se simultaneamente na preservação da privacidade e na

descentralização dos dados. As figuras 1 e 2 representam, respetivamente, o *market size* (Quota de mercado) por região e aplicação, proporcionando uma estimativa do potencial de adoção do produto por pessoas ou empresas numa determinada região ou aplicação específica [5], [6]. Importa sublinhar que, neste contexto, o próprio produto refere-se ao sistema de *Federated Learning* (FL).

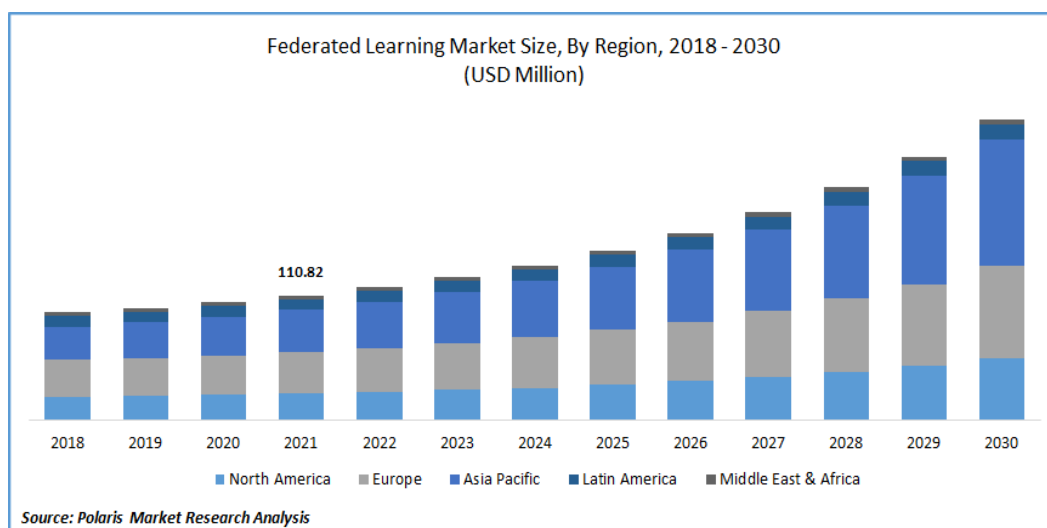


Figura 1 - Market size do FL, por região entre 2018 e 2030 (Projeção de valores entre 2023 – 2030 [5])

A figura 1, que representa a quota de mercado por região, proporciona uma visão abrangente do alcance global do FL. O crescimento observado sugere que esta abordagem tem vindo a ganhar terreno em várias regiões, indicando uma aceitação generalizada e reconhecimento das suas vantagens.

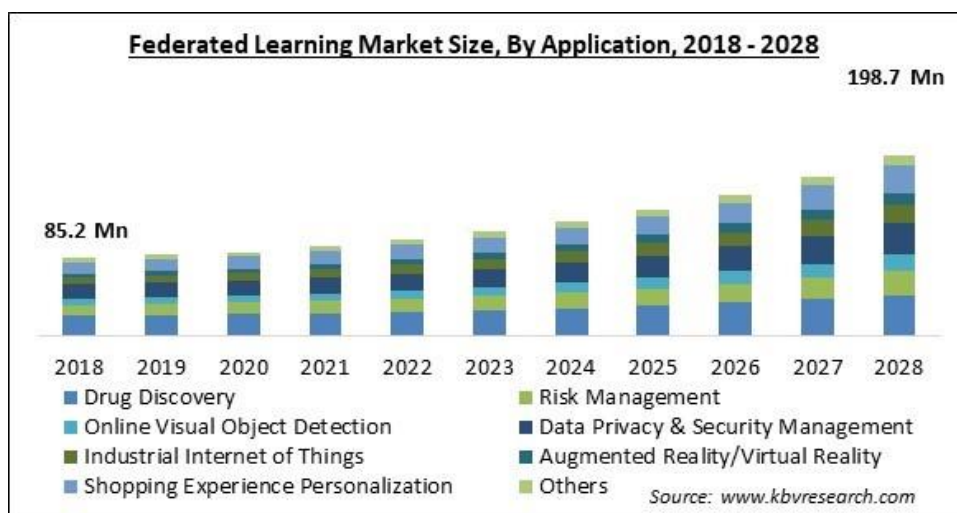


Figura 2 - Market size do FL, por aplicação entre 2018 e 2030 (Projeção de valores entre 2023 – 2028 [6])

Por sua vez, a figura 2, que representa a quota de mercado por aplicação, oferece uma análise mais específica das áreas onde o FL tem tido um impacto mais significativo. Esta segmentação por aplicação fornece insights valiosos sobre as preferências e tendências de adoção em setores específicos, orientando o entendimento do FL em contextos mais especializados.

Ao analisar as figuras 1 e 2, é possível observar o crescimento do *Federated Learning* (FL) nos últimos anos, indicando uma tendência positiva e uma aposta crescente por parte das empresas. O desenvolvimento constante do FL, evidenciado através destas representações gráficas, reflete não só um aumento significativo ao longo do tempo, mas também uma projeção otimista quanto à aceitação e implementação desta abordagem.

Importa notar que este crescimento evidenciado nas figuras 1 e 2 está correlacionado com o aumento do número de artigos científicos, livros, standards, entre outros em FL, como é possível observar pela figura 3. Os dados do gráfico foram obtidos com o auxílio da plataforma IEEE Xplore [7], através da filtragem temporal da documentação submetida relacionada com FL. Dado que a própria plataforma separava os diversos documentos em certas categorias, considerou-se apropriado adotar o seu princípio, daí ser exemplificado no gráfico o número de conferências, jornais, artigos, revistas, livros e standards. O aumento contínuo do número de publicações na área de FL, sugere um interesse contínuo por parte da comunidade científica em explorar e aprimorar os fundamentos e as aplicações do FL.

O constante desenvolvimento do FL, refletido tanto nas projeções positivas nas figuras 1 e 2 quanto no aumento na produção científica, reforça a ideia de que o FL está a tornar-se uma peça central no panorama do ML e da inteligência artificial. Este fenómeno não só destaca a eficácia do FL, mas também sugere que as organizações estão cada vez mais a reconhecer o seu potencial.

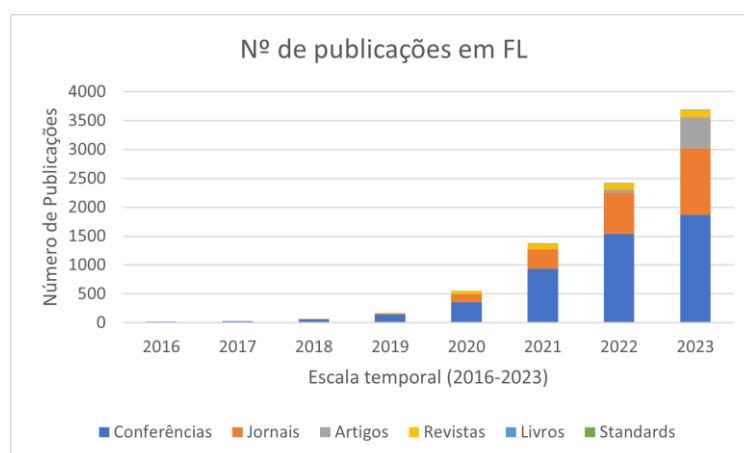


Figura 3 - Número de publicações que abordam o tema FL

1.2 Problema e objetivos

Esta dissertação tem como objetivo, analisar as condições ambientais em edifícios, nomeadamente em escritórios. Através de uma abordagem federada, pretende-se analisar e prever a evolução das condições de trabalho (temperatura, luminosidade, humidade, ruído, qualidade do ar e pressão), em gabinetes de um edifício de escritórios. Mais concretamente, o trabalho proposto, irá colecionar dados ambientais de três gabinetes distintos e prever a evolução da qualidade ambiental que depois poderá ser usada para melhorar algumas das funcionalidades do edifício, como por exemplo, o sistema HVAC (Heating, Ventilating and Air Conditioning) ou em português AVAC (Aquecimento, Ventilação e Ar Condicionado), a monitorização da qualidade do ar, o ajuste da iluminação e o acompanhamento dos níveis de ocupação.

Além da otimização do ambiente de trabalho num edifício de escritórios, esta dissertação visa responder até que ponto o FL é vantajoso em termos de escalabilidade, e eficiência energética quando comparada com as aprendizagens mais comuns, adotando um sistema baseado em aprendizagens distribuídas recorrendo ao uso de RaspBerry Pi. Neste sentido a seguinte pergunta de investigação foi considerada:

Pergunta de Investigação: Como poderá ser aplicada uma abordagem baseada em FL, de forma a melhorar as condições de trabalho em edifícios de escritórios, tendo por base condições de luminosidade, qualidade do ar, temperatura e níveis de ocupação?

Para responder à presente questão, a seguinte hipótese foi formulada:

Hipótese: Uma abordagem baseada em FL possibilita a previsão e otimização das condições ambientais em edifícios de escritórios, promovendo a descentralização do processamento de dados, explorando ainda o potencial de modelos de regressão e Deep Learning.

De forma a validar a hipótese, várias simulações serão feitas de forma a alcançar um ambiente otimizado tendo em conta os diversos fatores mencionados.

1.3 Abordagem e contribuições

A metodologia utilizada neste estudo desempenha um papel crucial na organização e execução dos processos e análises necessárias para atingir os objetivos propostos. Na figura 4, está delineado todo o processo a seguir de forma a assegurar uma abordagem sistemática ao problema proposto.

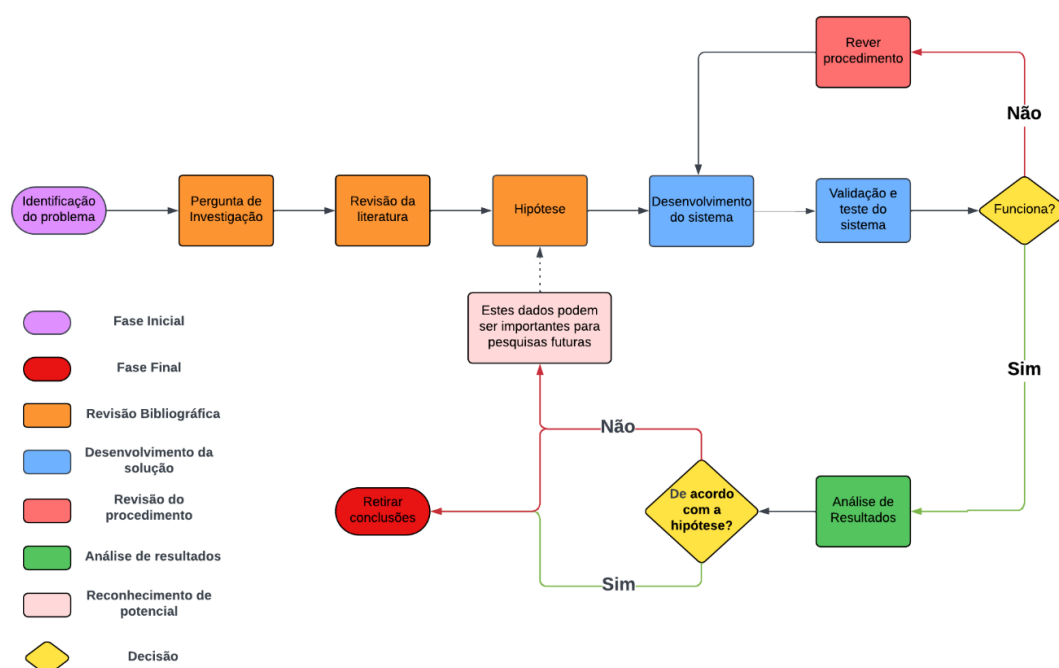


Figura 4 – Metodologia

A metodologia consiste dos seguintes passos:

1. **Identificação do problema:** Este passo consiste na identificação do problema que motivou a escrita deste documento. O problema foi identificado e descrito na secção 1.1, Contexto e Motivação.
2. **Pergunta de Investigação:** Este passo consiste na formulação clara e específica da pergunta de investigação que orientará todo o processo de investigação. Esta pergunta foi formulada na secção 1.2, Problema e Objetivos.
3. **Revisão da literatura:** Nesta etapa, é realizada uma revisão abrangente da literatura relacionada ao tema em estudo. Inclui fundamentos teóricos, bem como a análise de trabalhos e investigações anteriores. Esta etapa está representada no capítulo 2, Estado da Arte.
4. **Formulação da Hipótese:** Com base na revisão da literatura, este passo envolve a formulação da hipótese que será testada ao longo do processo. A hipótese foi formulada na secção 1.2, Problema e Objetivos.
5. **Solução proposta:** Após a formulação da hipótese é proposta uma solução ao problema já identificado. Este passo será abordado no capítulo 3.
6. **Desenvolvimento do sistema:** Neste passo será desenvolvido o sistema a testar. Notar que o desenvolvimento será abordado na próxima fase da dissertação, visto que esta é apenas a preparação.
7. **Validação:** Após o desenvolvimento, o sistema é submetido a processos de validação para garantir a sua eficácia e confiabilidade. Como no passo anterior, esta etapa não será para já abordado.
8. **Análise de Resultados:** Os dados originados pelo sistema são recolhidos e analisados. A análise destes tem como objetivo responder à pergunta de investigação e testar a hipótese formulada.
9. **Conclusões:** As conclusões finais dependem dos resultados obtidos, quer estejam ou não de acordo com a hipótese formulada. Neste passo são ainda mencionadas as limitações do estudo e possíveis direções para futuras investigações, além da publicação do mesmo.

1.4 Organização do documento

Este documento está separado em 6 capítulos, sendo estes os seguintes:

1. **Introdução:** Este capítulo apresenta a motivação e a pergunta de investigação.
2. **Estado da Arte:** Neste capítulo, é realizada uma análise abrangente do estado da arte na área de estudo. Inclui a fundamentação teórica relacionada com o estudo,

bem como a revisão crítica da literatura existente, desde estudos a métodos relevantes para o tema.

3. **Solução Proposta:** Este capítulo descreve a estrutura e implementação da solução, tanto em termos de hardware como de software. É explicado ainda o seu método de funcionamento.
4. **Análise de Dados:** Apresenta ainda a análise aos dados que serão usados para o treino dos algoritmos.
5. **Análise de Resultados:** Este capítulo contém a análise de todos os resultados obtidos.
6. **Conclusão:** Este capítulo apresenta as conclusões do estudo, bem como algumas ideias para futuras investigações.

ESTADO DA ARTE

Neste capítulo, será abordado de forma aprofundada o tema FL, explorando os seus princípios desde a sua definição até à contextualização histórica e aplicações. Este capítulo inclui uma secção sobre fundamentos teóricos inerentes ao IoT, que inclui uma taxonomia detalhada que se fundamenta em artigos científicos, proporcionando uma visão abrangente da estrutura do FL. Além disso, será dedicada uma secção ao estudo do FL no contexto da IoT, bem como as vantagens e desafios inerentes aos sistemas FL-IoT. É apresentada, uma revisão exaustiva de algoritmos de agregação de dados usados no âmbito do trabalho. Por último, é apresentado o trabalho relacionado que inclui, vários artigos científicos relevantes ao tema, solidificando assim a base teórica desta dissertação.

2.1 Princípios do FL

Nesta secção, é abordada a história do ML, é definido o conceito de FL bem como os benefícios, desafios e aplicações. Além disso será realizada uma comparação entre topologias centralizadas e descentralizadas com o FL. Na primeira subsecção será dada a definição do FL, bem como uma contextualização histórica do ML em geral, visto que o FL é uma tecnologia recente. De modo a facilitar a compreensão do desenvolvimento histórico do ML, estabeleceu-se uma escala temporal desde o final da década de 1940 até aos dias de hoje. Na subsecção seguinte, é abordada detalhadamente a comparação entre topologias centralizadas e descentralizadas com abordagens de FL, destacando algumas das diferenças entre elas. Nas 3 subsecções seguintes serão dados os benefícios, desafios e aplicações ligadas ao FL.

2.1.1 Definição e contextualização histórica

2.1.1.1 História do ML

1943 - 1º modelo de uma rede neuronal:

Warren McCulloch e Walter Pitts publicam o primeiro modelo matemático de uma rede neuronal, de forma a criar algoritmos para simular processos da mente humana [8].

1949 - Teoria Hebbiana:

Donald Hebb: Publicou o livro “*The Organization of Behavior: A Neuropsychological Theory*”, que apresenta uma teoria (teoria Hebbiana) com o objetivo de entender os mecanismos neuronais presentes na aprendizagem, memória e comportamento.

1950 - Teste de Turing

Alan Turing: Publicou “Computing Machinery and Intelligence”, [9], responsável pela introdução do teste de Turing e conceitos de computação, o que formou uma base futura para o desenvolvimento de ML e IA (Inteligência Artificial).

1951 - 1ª rede neuronal artificial (ANN – Artificial Neuronal Network):

Desenvolvimento da ANN, denominada por SNARC (Stochastic Neural Analog Reinforcement Calculator) que simulava uma rede de 40 neurónios.

1956 - Nascimento dos conceitos de IA e ML:

John McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon organizaram um workshop em Dartmouth, e é daí considerado o início conceptual da IA e ML [10]. Notar que embora já tivessem sido desenvolvidos modelos computacionais de redes neuronais, estes não eram associados a IA e ML pois os conceitos ainda não estavam formalmente estabelecidos.

1958 – Desenvolvimento do Perceptron:

Frank Rosenblatt, baseou-se no sistema nervoso dos humanos para construir uma máquina (Perceptron) que podia aprender através de dados [11]. Tornou-se a fundação para as redes neuronais modernas.

1959 – Designação do conceito de ML

Arthur Samuel, criou o conceito de ML e desenvolveu o primeiro algoritmo funcional do jogo de damas, marcando o início da inteligência algorítmica [12].

Desenvolvimento de sistemas especialistas baseados em regras (Rule-based Expert Systems)

– Abordagem dominante em AI nas décadas entre 1960-1980:

Edward Feigenbaum and Edward Shortliffe: Contribuíram significativamente para o desenvolvimento de sistemas especializados em regras.

Algoritmos de Machine Learning (1950s - 2000s):

Vladimir Vapnik e Alexey Chervonenkis: Pioneiros na introdução das Máquinas de Vetores de Suporte (SVMs – *Support Vector Machines*) na década de 1960. Ganharam popularidade na década de 1990, desempenhando um papel significativo na evolução do cenário dos algoritmos de classificação.

Surgimento de *Statistical Learning* (1990s - 2000s):

Leo Breiman: Introduziu as *Random Forests*, um método de *ensemble learning* (combina vários resultados das árvores de decisão de forma a alcançar um único resultado), que teve um impacto significativo na classificação e na regressão.

Trevor Hastie, Robert Tibshirani e Jerome Friedman, pioneiros do LASSO e das *boosted decision trees*, desenvolveram estratégias para aprimorar árvores de decisão fracas, transformando-as em modelos robustos e precisos. O impulso (*boost*), uma abordagem iterativa, corrige deficiências individuais das árvores, resultando em modelos mais precisos.

David Mackay e Christopher Bishop tiveram contribuições notáveis para os métodos Bayesianos em *Machine Learning*, introduzindo uma dimensão probabilística de forma significativa nos algoritmos.

Deep Learning (década de 2000 - presente):

Yann LeCun, Geoffrey Hinton e Yoshua Bengio: Foram pioneiros em técnicas de *Deep Learning* e contribuíram para o ressurgimento de redes neuronais. O *Deep Learning* tem sido aplicado em diversas áreas, desde reconhecimento de imagem e voz até ao processamento de linguagem natural.

Inteligência Artificial Contemporânea e Considerações Éticas (década de 2010 - presente):

A pesquisa em IA e ML expandiu-se para *Reinforcement Learning*, *generative adversarial networks* (GANs), entre outros. As implicações éticas e sociais da IA, incluindo justiça, responsabilidade e transparência, têm recebido mais atenção no desenvolvimento e uso de IA, em vez de se focar apenas no desempenho dos projetos.

Na figura 5 [13], está representada a linha temporal do ML desde a sua origem até 2018. Na figura encontram-se mais alguns eventos que não foram mencionados anteriormente.

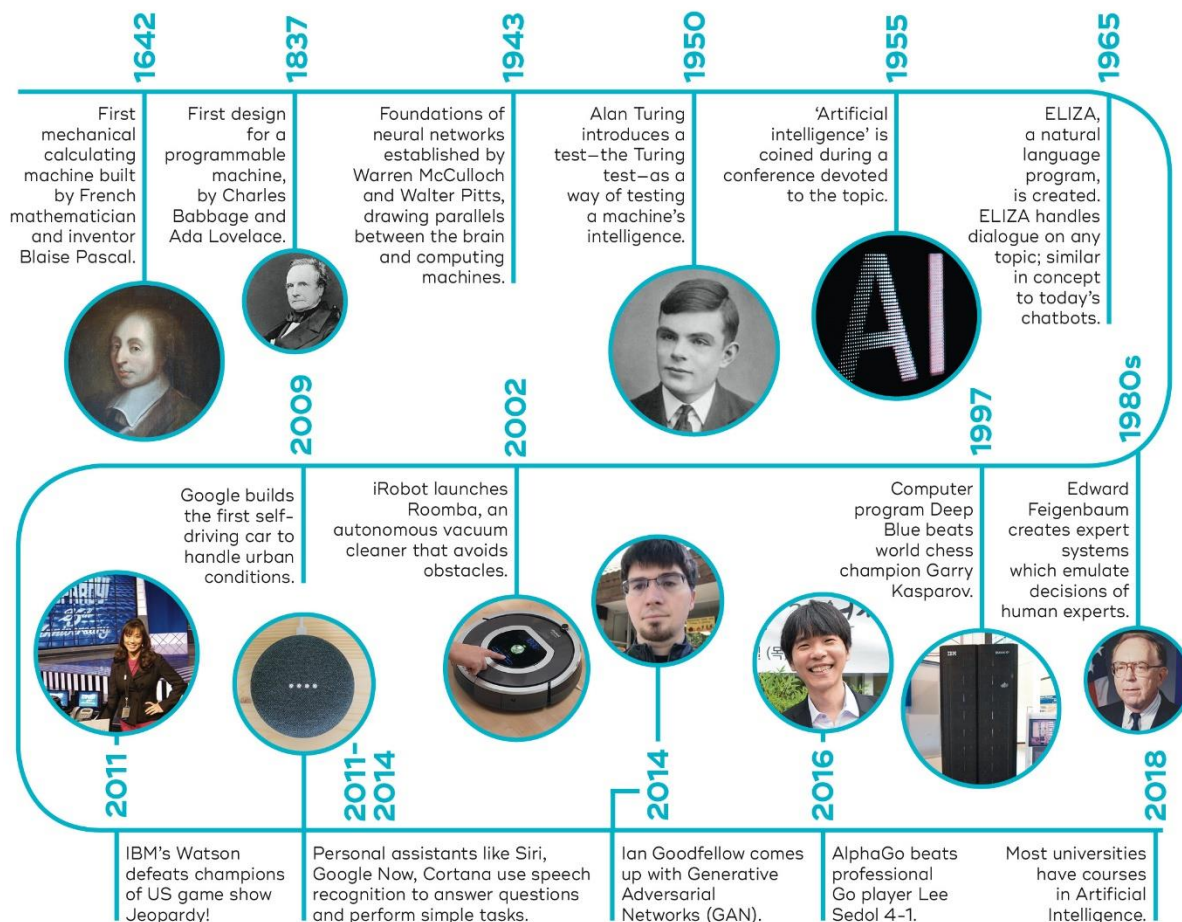


Figura 5 - Representação da linha temporal do ML [13]

2.1.1.2 Definição de FL

O conceito de FL foi introduzido em 2016 por McMahan [14]: “Designamos a nossa abordagem como FL, uma vez que a tarefa de aprendizagem é resolvida por uma federação flexível de dispositivos participantes (que designamos como clientes), coordenados por um servidor central”.

O FL representa uma configuração inovadora em ML, na qual diversos clientes, como dispositivos móveis, *edge devices* ou organizações, colaboram de maneira coordenada sob a orientação de um servidor central, geralmente um provedor de serviços. Nesse contexto, os dados de treino permanecem descentralizados, uma característica que diferencia o FL de abordagens mais tradicionais.

2.1.2 FL vs abordagens clássicas

No panorama das arquiteturas de *Machine Learning*, prevaleceram as topologias centralizadas, encapsulando tanto dados quanto modelos dentro de uma infraestrutura singular para coordenação e comunicação simplificadas. Embora operacionalmente eficiente, essa abordagem encontrou crescentes preocupações relacionadas à privacidade de dados, segurança e eficiência operacional.

A agregação de grandes quantidades de dados num único repositório eleva os riscos inerentes à privacidade nos modelos centralizados. Um simples ataque ao servidor central poderia expor ou pôr em causa um conjunto abrangente de dados, exigindo medidas rigorosas de segurança para proteção contra acessos não autorizados, especialmente ao lidar com informações sensíveis do utilizador.

Além disso, as topologias centralizadas enfrentam desafios relacionados aos custos associados à transferência de dados. Em cenários em que os dados precisam percorrer distâncias significativas para alcançar o servidor centralizado, os gastos com largura de banda e transferência de dados podem-se elevar. Este aspeto financeiro torna-se uma consideração crucial, especialmente em aplicações em larga escala, onde o movimento de dados é comum.

Adicionalmente, a centralização introduz desafios de latência, especialmente em aplicações em tempo real. O tempo de resposta (*delay*) associado à transferência de dados entre os *edge devices* e o servidor central pode resultar em atrasos inaceitáveis para operações sensíveis a prazos. Essa latência torna-se um fator crítico em aplicações como veículos autônomos, monitoramento de saúde e outros contextos em que a responsividade imediata é primordial [15].

Como resposta, o paradigma tem se deslocado para topologias descentralizadas, sendo estas baseadas num modelo distribuído em que os dados e a computação são divididos por vários nós. Essa descentralização serve como uma medida estratégica para mitigar os riscos à privacidade, assemelhando-se à distribuição de informações sensíveis por canais seguros, minimizando assim o potencial impacto de uma violação de segurança.

Simultaneamente, a relevância de *edge devices* e telemóveis tem-se intensificado. Esses dispositivos funcionam como centros de processamento localizados. A importância destes é acentuada no contexto da computação descentralizada, oferecendo a capacidade de executar tarefas próximas à fonte de dados, promovendo assim eficiência e responsividade.

Em essência, a evolução de arquiteturas centralizadas para descentralizadas, aliada à integração de *edge devices* e telemóveis, representa uma adaptação na aprendizagem de ML.

Ao contrário das metodologias distribuídas convencionais, o FL destaca-se pelos seus métodos no processo de treino e transferência de dados. Em vez de transferir dados brutos

entre os clientes e o servidor, cada cliente armazena localmente os seus próprios dados brutos e treina o seu modelo localmente. O que é compartilhado com o servidor são apenas os modelos e suas respectivas atualizações. Esta abordagem preserva a privacidade dos dados brutos de cada cliente, e ao mesmo tempo possibilita o desenvolvimento do modelo global.

O FL incorpora os princípios de *focused collection* e minimização de dados, ou seja [14], princípios que limitam a recolha de dados pessoais e focam-se nos dados mais importantes e necessários para resolver o problema em questão, podendo assim mitigar muitos dos riscos e custos de privacidade resultantes do *Machine Learning* tradicional e centralizado.

Em [16], foi feito um estudo de forma a comparar a convergência de resultados (eficácia), em três cenários distintos, aplicando diferentes abordagens de aprendizagem: centralizada, federada e descentralizada. Para os dados deste estudo foram usados dois conjuntos de dados *open-source*, MNIST e CIFAR-10. Dos resultados obtidos a abordagem que obteve pior desempenho foi a centralizada devido à quantidade de dados que devem ser transferidos para o servidor em cada ciclo de comunicação. Por sua vez a que obteve melhores resultados foi a federada devido aos métodos de treino e comunicação. A superioridade da abordagem federada foi consistente em todos os cenários, registando uma eficácia mínima de 86% no cenário 1 com o conjunto de dados CIFAR-10 e uma eficácia máxima de 97% nos cenários 2 e 3, utilizando o conjunto de dados MNIST. Em contraste, a abordagem centralizada apresentou uma eficácia mínima de 54% no cenário 1 com os dados CIFAR-10 e uma eficácia máxima de 73% no cenário 2 com os dados MNIST. Relativamente à abordagem descentralizada, os resultados variaram de uma eficácia mínima de 67% no cenário 1 com os dados CIFAR-10 a uma eficácia máxima de 81% no cenário 3 com os dados MNIST.

Baseado nestes resultados os autores afirmam que o FL é a melhor solução atualmente para todas as aplicações de Inteligência Artificial (IA), especialmente onde grandes conjuntos de dados e informações pessoais estão envolvidas, devido à proteção e privacidade dos dados, algo que outras abordagens não garantem.

Contudo, a abordagem ao FL nem sempre é a solução ideal para o problema. Devido à sua origem recente, a aplicação de modelos baseados em FL pode ter diversos problemas em termos de custo, eficiência, entre outros. Já existem diversas otimizações que se podem fazer para mitigar o impacto destes problemas, mas a sua aplicação pode aumentar mais ainda o preço do produto bem como a sua complexidade.

2.1.3 Benefícios do FL

O FL tornou-se crucial em contextos que lidam com dados sensíveis, onde a distribuição descentralizada dos modelos de treino é imperativa. Nesta subsecção, serão explorados alguns dos benefícios inerentes ao FL, destacando como esta abordagem se revela vantajosa em termos de privacidade, eficiência, segurança, entre outros.

- **Privacidade dos Dados** - A abordagem descentralizada da FL preserva a privacidade, uma vez que os dados sensíveis permanecem nos dispositivos locais. Neste contexto, apenas os parâmetros do modelo local são enviados ao modelo global, eliminando a necessidade de partilhar os dados referentes ao modelo local. Por outro lado, em abordagens centralizadas, onde a recolha dos dados pelo servidor central é total, a privacidade dos utilizadores não é garantida. Em [17], foi proposto um algoritmo neuronal federado baseado em Gated Recurrent Unit (GRU) denominado por FedGRU, com o objetivo de comparar o seu desempenho com outras abordagens, sendo estas GRU, SVM, LTSM e SAE. Notar que qualquer uma das abordagens, à exceção do FedGRU, poderá potencialmente comprometer a privacidade dos utilizadores. Os resultados indicaram que o desempenho do FedGRU foi bastante semelhante ao das outras abordagens, ao registar apenas mudanças mínimas na eficácia dos resultados, mas garantindo a privacidade total dos dados.
- **Redução da Sobrecarga de Comunicação** - Ao treinar modelos localmente e enviar apenas as atualizações para um servidor central, a FL reduz significativamente a carga na rede, tornando-a eficiente em termos de comunicação, especialmente em ambientes com largura de banda limitada.
- **Treino em Tempo Real em *edge devices*** - O FL permite o treino de modelos em *edge devices* em tempo real, possibilitando a aplicação em cenários como veículos autónomos, IoT e saúde, onde a resposta em tempo real é crucial.
- **Segurança** - O uso de abordagens FL, pode mitigar o risco de ciberataques, visto que não há transferência de dados ou informação na rede, apenas parâmetros necessários à atualização do modelo global. Já existem aplicações. Em [18], foi proposto um modelo de autenticação por voz denominado de FedUA, com o propósito de detetar e distinguir acertadamente 1251 utilizadores distintos. Os resultados indicaram uma eficácia de 80% (True Positive Rate) e uma taxa de 20% de falsos-positivos, resultando num desempenho melhor que outros modelos.
- **Eficiência energética** - O *Federated Learning* (FL) pode aumentar a eficiência energética dos algoritmos através da sua abordagem descentralizada. Ao permitir que o treino dos modelos ocorra localmente em dispositivos distribuídos, como smartphones ou

dispositivos IoT, evita-se a necessidade de transferir grandes quantidades de dados para um servidor central. Isto reduz a sobrecarga de comunicação e, ao realizar o treino nos dispositivos locais, é possível otimizar a utilização dos recursos de processamento disponíveis, contribuindo para uma melhor eficiência energética no processo de ML. Em [19], foi proposto um algoritmo de FL aplicado a redes sem fio da Internet das Coisas (IoT), com o intuito de minimizar o consumo de energia ou o tempo necessário para a conclusão do modelo. Os resultados indicam a eficácia do algoritmo FL proposto em comparação com abordagens tradicionais em redes IoT com recursos limitados.

No entanto, nem sempre os algoritmos de FL representam a solução ideal, pois apresentam desafios, especialmente no que diz respeito à escalabilidade e eficiência. Entre os principais obstáculos, os custos associados à comunicação e escalabilidade do sistema são os que exercem maior impacto no desenvolvimento desses sistemas [20].

2.1.4 Desafios ligados ao FL

Comunicação: A necessidade dos dados serem mantidos localmente, devido às preocupações com a privacidade e segurança dos mesmos, implica que para uma rede com bastantes aparelhos a comunicação poderá ficar mais lenta devido aos recursos limitados que a compõem, como por exemplo a largura de banda. É então importante desenvolver métodos de comunicação mais eficientes, que reduzam ou o número de ciclos de comunicação ou até mesmo o tamanho das mensagens enviadas [21].

Heterogeneidade de dados e dispositivos: A heterogeneidade de dados em redes federadas nem sempre é tida em conta, assumindo sempre que conjuntos de dados locais são semelhantes, o que nem sempre acontece. As distribuições heterogêneas de dados entre dispositivos podem ter impacto no desempenho do modelo global. Esse impacto é atribuído a diferenças nas capacidades de armazenamento, computação e comunicação de cada dispositivo, devido a variabilidades em hardware, conectividade de rede (3G, 4G, 5G e Wi-Fi) e nível de energia da bateria. Além disso, fatores como o tamanho da rede e restrições relacionadas aos sistemas, resultam em que apenas uma pequena fração dos dispositivos estão ativos ao mesmo tempo, o que piora desafios como mitigação de *stragglers* (nó ou dispositivo que é significativamente mais lento que os outros, o que causa atrasos na computação geral) e tolerância a falhas, especialmente quando dispositivos ativos podem falhar durante uma iteração devido a limitações de conectividade ou energia. O *Federated Learning* deve então antecipar um número baixo de

participantes, tolerar hardware heterogêneo e deve ser robusto o suficiente para garantir o seu funcionamento em caso de falhas em dispositivos [21].

Privacidade: O FL protege os dados de dispositivos ao compartilhar atualizações do modelo, mas a comunicação dessas atualizações pode comprometer a privacidade. Certos dispositivos maliciosos, que podem atuar como clientes honestos, podem influenciar o desempenho do modelo através de atualizações incorretas. Este evento denomina-se por *model poisoning*. Métodos recentes visam melhorar a privacidade, mas frequentemente resultam numa redução de desempenho ou eficiência do sistema. É então importante fazer certos compromissos, entre privacidade e desempenho, de forma a alcançar o melhor modelo possível [21].

Custo: Idealmente e possivelmente no futuro a abordagem ao FL deverá ser das tecnologias mais eficientes em termos de custo, quando comparada com outras abordagens mais tradicionais. No entanto, atualmente os custos ligados às atualizações, dispositivos da rede e até da própria comunicação podem ser um desafio para empresas que as usam [21].

Classificação e limpeza de dados: O FL, apesar de proteger a privacidade, limita o acesso aos dados brutos dos utilizadores. Ou seja, dados que estejam em falta ou que sejam irrelevantes, não podem ser eliminados pois não são detetados [21].

Escolha do cliente: A escolha dos clientes para modelos de FL é de extrema importância, revelando-se também como um grande desafio. Notar que neste contexto, dá-se o nome de cliente a um qualquer dispositivo integrado no sistema, por exemplo *edge devices*. Segundo [22], na consideração dos clientes devem ser tidos em conta quatro fatores, heterogeneidade dos dispositivos (já falado anteriormente), capacidade de adaptação dos dispositivos, mecanismo de incentivo e agregação adaptativa. Como a heterogeneidade já foi mencionada, apenas serão mencionados neste ponto os restantes fatores, sendo estes os seguintes [21]:

- **Capacidade de adaptação dos dispositivos:** A capacidade dos dispositivos em ajustar o seu comportamento com base na sua participação no modelo é uma característica fundamental. Dois aspetos cruciais nesta adaptação são a participação flexível e as atualizações parciais. A participação flexível refere-se à capacidade de os dispositivos determinarem o seu envolvimento no modelo, permitindo-lhes escolher a quantidade de dados a enviar. Por sua vez, as atualizações parciais possibilitam a transmissão de atualizações parciais em vez de uma atualização completa, reduzindo significativamente os custos de comunicação associados.
- **Mecanismo de incentivo:** Incentivar os clientes a compartilhar os seus modelos é essencial para melhorar o desempenho e a escalabilidade do sistema, contudo nem

sempre é fácil oferecer os incentivos certos dado que dependem de cliente para cliente. Segundo [22], existem três métodos de incentivo bastante comuns:

- **Incentivo Monetário:** Recompensa os clientes monetariamente pelas suas contribuições;
- **Incentivo por Reputação:** Os clientes que contribuem ativamente e fornecem atualizações de qualidade são recompensados com uma maior reputação.
- **Incentivo por Tokens:** Semelhante ao monetário, este método baseia-se na recompensa com *tokens* que podem ser usados em serviços adicionais.
- **Agregação Adaptativa:** O propósito desta agregação tem como objetivo ajustar a quantidade de dados transferida para o servidor, considerando a largura de banda dos dispositivos. Ou seja, dispositivos com menor capacidade computacional ou com problemas de conectividade podem enviar uma quantidade inferior de dados, enquanto dispositivos com melhor desempenho podem transmitir dados a uma taxa superior.

A implementação em larga escala do FL encontra-se condicionada por diversas barreiras. A falta de padronização na sua implementação dificulta a interoperabilidade entre sistemas e setores, sendo crucial o desenvolvimento de padrões comuns. As questões de privacidade persistem, apesar do FL preservar a privacidade dos dados individuais, especialmente em setores sensíveis como saúde e finanças. A descentralização do treino do modelo introduz desafios de segurança, exigindo medidas robustas para proteger os dados durante a transmissão e no armazenamento local dos dispositivos. A eficácia do FL depende de uma conexão confiável entre os dispositivos locais e o servidor central, o que pode ser prejudicada em regiões com rede instável. A complexidade técnica é uma consideração adicional, já que a implementação em larga escala necessita frequentemente de ajustes técnicos significativos nas infraestruturas existentes, o que pode ser complexo e requerer investimentos substanciais. Os custos associados à implementação, como atualizações de infraestrutura e comunicação, podem representar uma barreira, especialmente para organizações mais pequenas.

2.1.5 Aplicações de FL

FL é essencial para certas aplicações que lidam com dados sensíveis em que os seus modelos de treino são distribuídos. Algumas destas são expressas abaixo:

IoT – Nos dias de hoje certos equipamentos ou aparelhos, requerem a recolha de dados em tempo real de forma a reagir às circunstâncias que possam surgir, atualizando o modelo

também em tempo real. São exemplos destes aparelhos: sensores, casas inteligentes e veículos autónomos;

Área da saúde – Devido às restrições impostas por diversos regulamentos, a partilha de informações de saúde dos utentes torna-se desafiadora. A abordagem ao FL, potencia uma resposta ao “problema”, em que passa a ser possível a construção de modelos sem que os dados em causa sejam ligados ao utente. Estes modelos poderão servir para a deteção e previsão de doenças bem como na investigação médica;

Reconhecimento facial – O treino dos modelos de reconhecimento facial, pode ser feito através da recolha direta de dados ligados a um certo utilizador. Mesmo que o modelo atinja o desempenho pretendido, este pode estar a usar dados sensíveis para o qual pode não ter autorização, sendo que cabe ao utilizador permitir ou recusar a partilha dos seus dados. A abordagem do FL resolve o problema construindo os modelos sem expor os dados sensíveis [23];

Fraude fiscal – Em modelos de ML convencionais é difícil detetar fraudes fiscais, pois as próprias instituições fiscais não são autorizadas a partilhar certos dados devido à privacidade e segurança dos mesmos. Em [24] foi proposto um modelo de deteção de fraude baseada em FL, de modo a combater fraude com cartões de multibanco;

Smart Grid – O desenvolvimento contínuo das redes elétricas representa um passo crucial na procura por sistemas mais eficazes, resilientes e inteligentes. Em particular o desenvolvimento das redes inteligentes (*Smart Grids*), tem revolucionado a forma de como consumimos energia (no caso do consumidor), através da incorporação de diversos dispositivos de medição, automação e comunicação, de forma a alcançar o melhor desempenho possível. O FL surge como abordagem promissora, ao introduzir vantagens na eficiência e segurança das redes inteligentes, devido ao treino local dos modelos, inexistência de transferência de dados sensíveis entre dispositivos e adaptação local do modelo às particularidades locais. Em [25], foi proposto um modelo de FL de AIoT (*Artificial Internet of Things*), com o propósito de garantir a transferência de dados de forma segura e eficiente em redes inteligentes entre o cliente e o operador da rede. Os resultados obtidos revelaram que o modelo permitiu a segurança na transferência dos dados do cliente, permitiu que o operador mesmo só tendo acesso a certos dados conseguisse um aumento na receita e reduziu a latência na comunicação. Esta abordagem não só fortalece a segurança dos dados, mas também demonstra o potencial que tem na otimização das redes inteligentes;

Veículos autónomos – O número de veículos com características autónomas, como *cruise control* e sistema de apoio para o estacionamento tem vindo a aumentar globalmente. As empresas que comercializam estes veículos recolhem dados dos mesmos, alguns dos quais podem

ser privados como os lugares que visita, devido ao constante mapeamento da localização do veículo. Em [26], foi proposto um modelo de FL com o intuito de treinar a detecção de objetos na via pública, bem como comparar o seu desempenho com outras abordagens tradicionais centralizadas, sem a necessidade de partilhar qualquer tipo de dados entre entidades. Os resultados obtidos foram positivos, assegurando tanto a privacidade e segurança dos dados bem como um bom desempenho sem a necessidade de um algoritmo centralizado de ML de alto desempenho;

Monitoramento Ambiental – A monitorização ambiental é crucial para a compreensão e preservação do ecossistema, sendo essencial para avaliar a qualidade do ar, dos recursos hídricos (por exemplo água) e da biodiversidade. O FL, ao descentralizar o processo de ML, destaca-se como uma abordagem inovadora para aprimorar esta monitorização. Esta técnica permite uma colaboração eficiente entre diferentes dispositivos, como sensores e estações meteorológicas, sem a necessidade de partilhar dados sensíveis. Além de proporcionar privacidade, o FL permite aos engenheiros atuar com maior rapidez devido às atualizações em tempo real e adaptação do modelo às condições locais onde está inserido o sensor. Em [27], foi apresentada uma aplicação de monitorização da qualidade da água com recurso a uma abordagem de FL, que permitiu decisões mais rápidas por parte dos engenheiros impedindo que a água poluída chegue aos consumidores.

2.2 Fundamentos Teóricos

Nesta secção, são explorados os fundamentos do FL, começando pela sua taxonomia, que classifica as diferentes arquiteturas e técnicas de colaboração entre dispositivos. Em seguida, é abordado o papel das séries temporais, frequentemente utilizadas em aplicações de FL para a análise de dados sequenciais. A integração da FL no contexto da Internet das Coisas (IoT) é discutida, destacando os desafios de comunicação e processamento em dispositivos com recursos limitados. São também apresentados os principais algoritmos e modelos de FL, focando nas técnicas de agregação e optimização distribuída. Por fim, faz-se uma breve revisão dos trabalhos relacionados que fundamentam este estudo.

2.2.1 Taxonomia do FL

De forma a compreender a estrutura do FL, é essencial explorar todas as suas abordagens, desde a partição de dados, aos modelos usados, bem como as várias arquiteturas de comunicação. A taxonomia usada será baseada em [28]. Esta subsecção apresentará uma visão abrangente da taxonomia do FL, focando-se em aspetos críticos relacionados com modelos de aprendizagem ML, mecanismos de privacidade, dimensão e motivação da federação, arquitetura e partição de dados.

2.2.1.1 Partição de Dados

Com base na forma como os dados são distribuídos nos espaços de amostras e de características, os FLS (*Federated Learning Systems*) podem ser tipicamente categorizados em horizontais, verticais e híbridos.

Horizontal

No HFL (*Horizontal Federated Learning*) diferentes conjuntos de dados compartilham características semelhantes, mas têm pouca sobreposição nas amostras. É útil quando os clientes têm dados semelhantes, mas não idênticos, permitindo o treino local e a partilha de atualizações do modelo para construir colaborativamente um modelo global. Isso ocorre sem a necessidade de enviar dados brutos centralmente, promovendo a privacidade dos dados [28]. Em [29], foi apresentado um cenário que envolve dois bancos, destacando a pouca sobreposição nos dados dos utilizadores, embora as características dos dados recolhidos sejam praticamente as mesmas. A figura 6 retirada de [30], ilustra graficamente este método de partição dos dados.

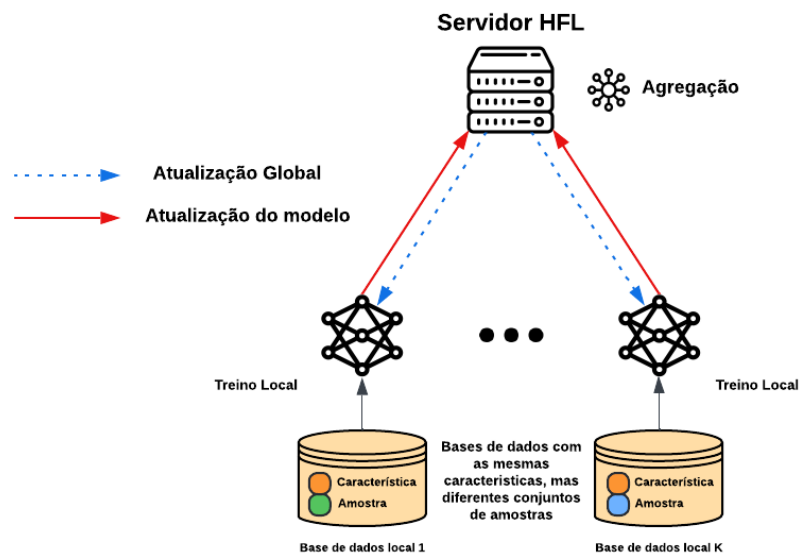


Figura 6 - Representação gráfica do HFL (adaptada de [30])

Vertical

No VFL (*Vertical Federated Learning*) diferentes conjuntos de dados apresentam sobreposição em algumas amostras, mas diferem nas características. É útil quando os clientes têm informações distintas, mas as amostras são semelhantes, o que torna possível a colaboração entre clientes na construção de modelos sem a necessidade de compartilhar todos os dados. De forma a recolher as amostras sobrepostas, são geralmente usadas técnicas de *entity alignment* (processo pela qual as amostras de cada conjunto de dados são identificadas e mapeadas). De forma a proteger a privacidade dos dados, o treino do modelo é feito recorrendo a métodos de encriptação [28]. Em [29], foi apresentado um cenário que envolve um banco e uma plataforma de e-commerce, evidenciando as distintas características recolhidas, como o comportamento financeiro do utilizador no banco e o histórico de compras na plataforma de e-commerce. Mesmo com sobreposição de utilizadores em ambas as entidades, é perceptível a diferença nos dados recolhidos. Na figura 7 retirada de [30], está exemplificado graficamente este método de partição de dados.

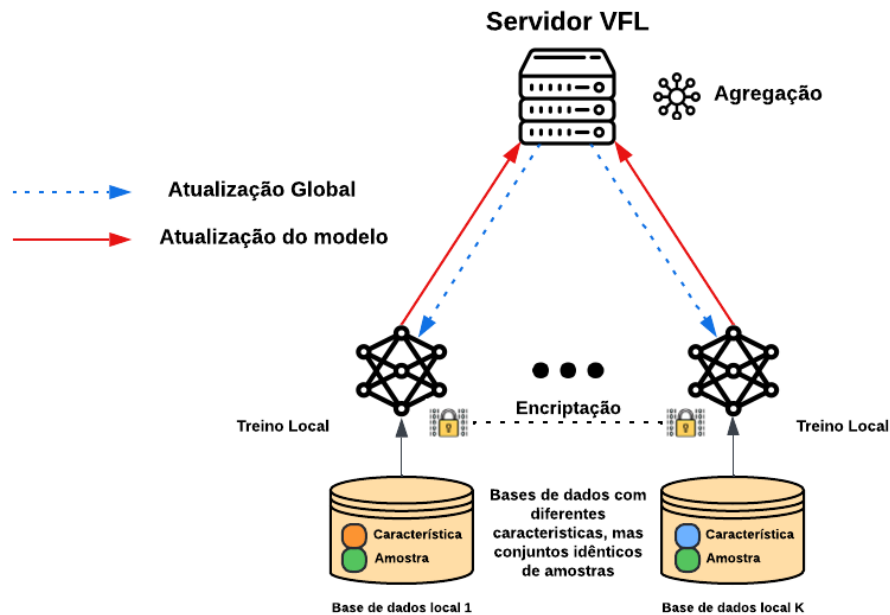


Figura 7 - Representação gráfica do VFL (adaptada de [30])

Híbrida

Em diversas aplicações, embora os atuais FLSs se concentrem predominantemente num tipo específico de divisão, a partição dos dados entre clientes pode envolver uma combinação de abordagens, sendo esta uma junção entre partições horizontal e vertical. Um cenário concreto foi dado em [28], onde se considera um sistema de diagnóstico de cancro. Em que um grupo de hospitais pretende construir um FLS, para o diagnóstico de cancro, mas cada hospital tem um conjunto distinto de pacientes, bem como resultados médicos diferentes. Proposto por [31], o FTL (*Federated Transfer Learning*) surge como uma possível solução para esse desafio. Este sistema de FTL é concebido para aprender uma representação entre as características dos clientes, sendo que, no contexto deste exemplo, os clientes são representados pelos hospitais. O sistema utiliza amostras partilhadas entre hospitais, o que significa que, ao utilizar dados que são comuns entre diferentes hospitais, é possível aprender uma representação unificada dos dados recolhidos. Essa representação é projetada para ser relevante para todos os hospitais, mesmo considerando as diferenças nos pacientes e resultados dos exames. Na figura 8 retirada de [30], está exemplificado graficamente este método de partição de dados.

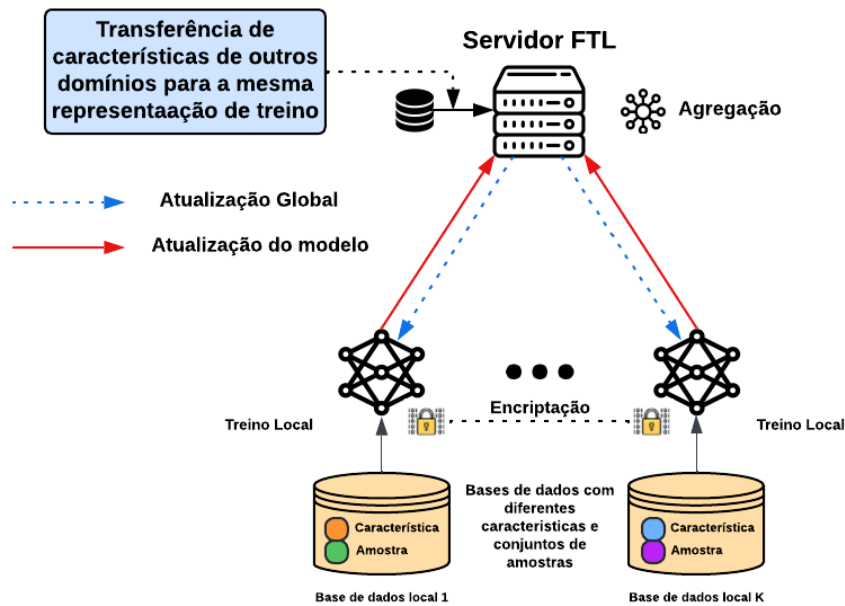


Figura 8 - Representação gráfica do FTL ou Hybrid FL (adaptada de [30])

2.2.1.2 Modelos de ML

No FL, os participantes colaboram para treinar modelos de ML avançados, destacando-se a utilização de redes neurais (NN – Neural Network), onde é frequente o uso do algoritmo SGD (*Stochastic Gradient Descent*) no treino das mesmas. O modelo de NN destaca-se principalmente nas áreas de classificação de imagem, previsão de palavras, entre outros.

Árvores de decisão são amplamente utilizadas devido à eficiência e fácil interpretação em comparação com NNs. FLS baseados em árvores de decisão, como GBDTs (Gradient Boosting Decision Trees), destacam-se pelo excelente desempenho em classificação e regressão.

Para além destes modelos, os modelos lineares (regressão linear, logistic regression e SVM – Support Vector Machine) são facilmente usados, pois quando comparados com os NNs são bem mais fáceis de entender.

2.2.1.3 Mecanismos de privacidade

Em FL, embora os dados locais não sejam expostos, os parâmetros do modelo trocados podem revelar informações sensíveis, tornando-se vulneráveis a ataques como *model inversion* e *membership inference*. Recorre-se a mecanismos de privacidade, como criptografia

homomórfica e privacidade diferencial, para proteger os dados. Métodos criptográficos, como criptografia homomórfica e SMC (*Secure Multiparty Computation*), protegem a privacidade, embora implique uma sobrecarga computacional devido à constante encriptação e desencriptação dos dados. A privacidade diferencial introduz ruídos para proteger dados ou parâmetros dos modelos dos participantes, mas pode resultar em modelos menos precisos.

Diversos métodos podem ser combinados para melhorar ou garantir a privacidade, incluindo o uso de ambientes de execução confiável (TEE - *Trusted Execution Environment*) e modelos de ameaças variáveis durante o FL, pois os ataques podem ocorrer em qualquer fase do processo, desde a introdução dos dados até à partilha do modelo após a conclusão. Destacam-se ataques comuns, como *poisoning* de dados, modelos, falhas bizantinas e inferência. Para uma compreensão mais aprofundada, segue-se uma breve explicação desses ataques, entre outros, bem como alguns mecanismos de proteção já mencionados [28]:

Ataques:

- **Poisoning de dados e do modelo** – Clientes ou participantes maliciosos, comportam-se como clientes honestos, podendo influenciar o treino do modelo ao enviar dados incorretos [32].
- **Falhas Bizantinas** – Representa uma situação em que um ou mais componentes de um sistema computacional, especialmente em sistemas de computação distribuída, enfrentam falhas, resultando na falta de informações precisas sobre a falha de um componente ou na veracidade das informações do sistema [33].
- **Inferência (*inference*)** – Utilizado para deduzir se um conjunto de dados pertence ou não a um modelo de treino. Este tipo de ataque é usado quando é feita a atualização ao modelo global através da transferência dos parâmetros do modelo local [32].
- **Model Inversion** – Inverte o processo de modelação do modelo de forma a deduzir informações ou parâmetros, através de *queries* ao modelo utilizando os dados de output [34].

Mecanismos de defesa:

- **Criptografia Homomórfica** – Técnica que permite a transformação de dados num texto cifrado (*ciphertext*), mantendo a capacidade de análise e manipulação como se estivesse na sua forma original. Além disso, possibilita a realização de operações matemáticas complexas nos dados encriptados sem comprometer a segurança da encriptação [35].
- **Privacidade Diferencial** – Abordagem matemática para proteger os indivíduos quando os seus dados são utilizados em conjuntos de dados [36]. Muitos estudos

usam esta abordagem de modo a garantir que os participantes não tenham conhecimento sobre a participação de outros participantes no processo de aprendizagem [28].

- **Secure Multiparty Computation** – Garante que todos os participantes não podem aprender nada a não ser o output. Pode ser usado na agregação segura dos gradientes transferidos [28].

2.2.1.4 Arquitetura de Comunicação

Existem duas abordagens principais na comunicação em FLSs, centralizada e descentralizada, sendo estas exemplificadas nas figuras 9 e 10, respetivamente.

Na abordagem centralizada a comunicação é centrada no servidor, visto que é este que define quais vão ser as atualizações feitas no modelo global através da recolha dos modelos locais dos dispositivos participantes (como é no caso do Google Keyboard).

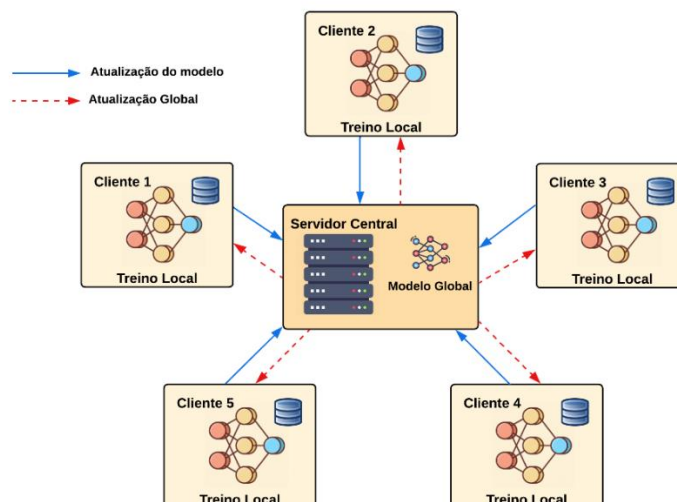


Figura 9 - Topologia centralizada (adaptada de [30])

Enquanto na abordagem descentralizada os participantes podem comunicar entre si, em que cada um deles pode atualizar diretamente os parâmetros globais [28].

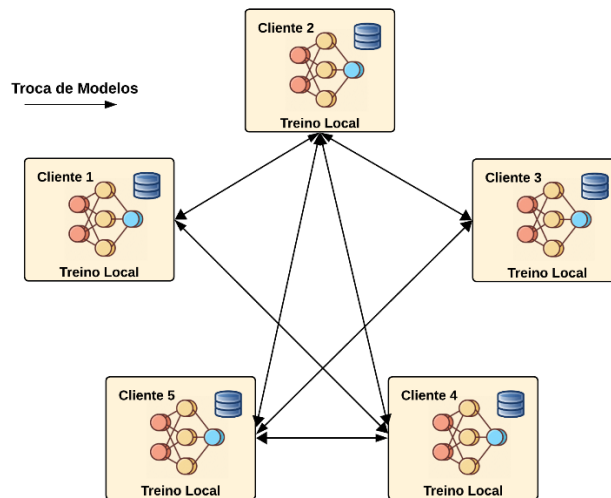


Figura 10 - Topologia descentralizada (adaptada de [30])

Embora o design centralizado seja amplamente utilizado nos estudos existentes, o design descentralizado é preferido em alguns aspectos, uma vez que concentrar informações num único servidor pode trazer riscos na segurança e privacidade dos dados, como já foi mencionado anteriormente. No entanto, o design da arquitetura de comunicação descentralizada pode ser complexo, devendo ter em consideração a sobrecarga de comunicação.

Segundo [28], existem três tipos de implementações da abordagem descentralizada, sendo eles os seguintes:

- **P2P (Peer-to-Peer)** – Todos os participantes têm os mesmos privilégios e são tratados de forma igual. O princípio deste sistema é agregar os recursos presentes nos *edge devices* e partilhá-los entre participantes [37]. Um exemplo dado que implementa o P2P é o SimFL.
- **Grafo** - A arquitetura de comunicação é modelada como um grafo, considerando restrições adicionais como latência e tempo de computação. Em [38], foi proposto um algoritmo para encontrar uma topologia de comunicação que otimize o desempenho do sistema, garantindo eficiência na colaboração distribuída.
- **Blockchain** - Proporciona uma plataforma descentralizada para armazenar informações relacionadas à aprendizagem federada. Essa abordagem visa garantir a transparência do processo, aproveitando as características de segurança e imutabilidade oferecidas pela tecnologia de blockchain.

2.2.1.5 Escala de Federação

A escala de federação refere-se à extensão ou magnitude do FLS, especificamente em termos do número de participantes e da quantidade de dados armazenada em cada participante. Ou seja, classifica os FLSs com base no seu tamanho, considerando se os participantes são grandes organizações ou centros de dados (cross-silo) ou inúmeros dispositivos mais pequenos (cross-device). A classificação destes sistemas pode ser dividida em dois tipos [28][39]:

- **Cross-device FL** – Os participantes são entidades pequenas distribuídas com poucos dados no seu armazenamento. Para este tipo de sistema são necessários inúmeros dispositivos que possam participar no processo de aprendizagem, o que levanta certas preocupações quanto ao consumo de energia, e por isso os dispositivos não podem ser sujeitos a tarefas de treino complexas. Normalmente os participantes são dispositivos móveis. Notar que neste caso o sistema deve ser capaz de gerir todos estes dispositivos, bem como os possíveis problemas na comunicação entre os dispositivos e o servidor. Um exemplo aplicacional deste sistema é o Google Keyboard.
- **Cross-silo FL** – Os participantes, neste contexto, são frequentemente empresas ou organizações, como hospitais e bancos. O número de participantes é limitado, e espera-se que cada um deles contribua para o processo de treino. Alguns desafios associados ao modelo cross-silo envolvem a eficácia e eficiência, provenientes da heterogeneidade entre os participantes e da distribuição de dados sob restrições de privacidade.

2.2.1.6 Motivação da Federação

Como já foi mencionado anteriormente, em aplicações reais de FL, os clientes precisam de motivação para participar. Isso pode ser impulsionado por regulamentações ou incentivos. A participação é voluntária, mas os clientes são beneficiados com uma maior precisão nos modelos. Em setores como saúde, hospitais podem usar FL para treinar modelos sem compartilhar dados sensíveis. O Google Keyboard permite aos utilizadores escolherem participar para melhorar a previsão de palavras. Desenvolver mecanismos de incentivo justos é extremamente complexo, mas é essencial no incentivo das empresas de modo a conseguir contribuições significativas [28].

2.2.2 Séries Temporais

As séries temporais (*time series*) são um conjunto de observações ordenadas cronologicamente, geralmente espaçadas em intervalos de tempo uniformes, como segundos, minutos, horas, dias, meses ou anos, como é o caso das séries usadas nesta dissertação. Neste subcapítulo serão abordadas as suas características, alguns métodos de análise, bem como as suas aplicações e requisitos para a sua análise. Para o âmbito da dissertação, trabalhar com séries temporais é fundamental para identificar tendências e padrões nos dados de temperatura e outros sensores, detetar anomalias e comportamentos fora do normal, bem como prever futuras leituras com base em dados históricos. Além disso, a análise de séries temporais permite contextualizar as variações nos dados, avaliar a estabilidade e a precisão dos sensores, e compreender melhor as dinâmicas do sistema monitorizado.

2.2.2.1 Componentes das séries temporais

Para compreender e analisar séries temporais de forma eficaz, é fundamental identificar os seus componentes principais. Cada série temporal pode apresentar diversos padrões e comportamentos que afetam a análise e interpretação dos dados. As séries temporais apresentam 4 componentes principais [40]:

- **Tendência:** Representa um padrão subjacente que pode estar a crescer, decrescer ou a manter-se constante ao longo do tempo. Pode ser linear ou não linear.
- **Sazonalidade:** Refere-se a padrões que se repetem em intervalos regulares de tempo, como estações do ano, meses ou dias da semana.
- **Ciclos:** Semelhantes à sazonalidade, mas não têm um período fixo e podem ocorrer em intervalos irregulares devido a fatores económicos, sociais ou outros.
- **Componente Residual:** A componente residual representa o ruído ou a variação aleatória nos dados após a remoção das tendências e sazonalidades. Esta componente captura as flutuações que não podem ser explicadas pelos outros componentes.

2.2.2.2 Análise de séries temporais

A análise de séries temporais é uma área importante da estatística e da análise de dados, utilizada para compreender, modelar e prever comportamentos de dados ao longo do tempo. Esta envolve diversas técnicas e métodos para examinar dados ao longo do tempo, identificando padrões, tendências e sazonalidades. Alguns dos métodos mais comuns incluem [40]:

- **Decomposição:** Separar a série temporal em componentes de tendência, sazonalidade e ruído.

- **Modelos ARIMA:** (AutoRegressive Integrated Moving Average) são usados para modelar séries temporais estacionárias, incluindo componentes de autorregressão, diferenciação e média móvel.
- **Modelos Exponenciais Suavizados:** Técnicas como a suavização exponencial simples, dupla e tripla (Holt-Winters) são usadas para prever séries temporais com tendência e sazonalidade.
- **Redes Neurais e Modelos de Deep Learning:** Modelos como LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit) são aplicados na previsão de séries temporais complexas.

Um conceito fundamental na análise de séries temporais é estacionariedade. Esta refere-se à condição em que as propriedades estatísticas, como a média e variância, permanecem constantes ao longo do tempo [40]. Muitas das técnicas e modelos utilizados assumem que a série é estacionária, tornando essencial a verificação e, se necessário, a transformação da série de forma a garantir essa condição. Na subsecção seguinte serão mencionados alguns dos requisitos destas séries temporais, bem como se pode avaliar a estacionariedade das séries temporais.

2.2.2.3 Requisitos para a análise de séries temporais

De forma a realizar uma análise eficaz de séries temporais, são necessários alguns requisitos essenciais:

- **Conjunto Extenso de Dados:** Um grande número de dados é necessário para garantir consistência e fiabilidade. Um conjunto de dados extenso assegura uma amostra representativa e permite filtrar dados ruidosos.
- **Dados Consistentes:** É importante que os dados sejam recolhidos em intervalos de tempo uniformes para facilitar a análise e interpretação dos padrões.
- **Tratamento de Dados Perdidos:** Deve-se lidar com dados inexistentes ou intervalos de tempo irregulares, utilizando técnicas de interpolação ou preenchimento.
- **Pré-processamento de Dados:** Muitas séries temporais não são estacionárias, exigindo técnicas avançadas de pré-processamento, como diferenciação ou transformação de dados.
- **Ferramentas de Análise:** Utilização de software e ferramentas estatísticas adequadas para modelar e prever séries temporais, como R, Python (com bibliotecas como pandas, statsmodels e scikit-learn) e outras plataformas de análise de dados.

Para o caso desta dissertação a maior parte dos requisitos já foram preenchidos, desde um conjunto extenso de dados, consistência dos mesmos, pois são recolhidos e tratados em intervalos de tempo uniformes e é usada a linguagem Python para a análise das séries com o uso de bibliotecas como o pandas, statsmodels e scikit-learn. Falta apenas garantir a estacionariedade desta série. No contexto desta dissertação, a série temporal utilizada não é uma série temporal simples, mas sim uma série temporal multivariada. Uma série temporal simples envolve apenas uma variável ao longo do tempo. Por exemplo, uma série temporal simples poderia ser a temperatura registada diariamente ao longo de um ano, pois envolve apenas a variável temperatura. Para o caso desta tese, como se está a avaliar a qualidade do ambiente que depende de 5 variáveis, denomina-se por série temporal multivariada [41]. Trata-se de uma série com um único timestamp com 5 variáveis diferentes. De forma a testar a estacionariedade da série como um todo, recorreu-se a um teste Augmented Dickey-Fuller (ADF), que determina a estacionariedade de uma série temporal. No caso de uma série multivariada, este teste deve ser aplicado a cada variável de forma independente.

O teste ADF, tem em conta duas hipóteses:

- **Hipótese Nula (H_0):** A série temporal não é estacionária.
- **Hipótese Alternativa (H_1):** A série temporal estacionária.

O teste aplica uma regressão sobre os valores da série temporal, ao tentar verificar a presença de uma raiz unitária, dado que esta implica um comportamento de não estacionariedade. Se uma série temporal possui uma raiz unitária, implica que a série temporal não retorna um valor médio constante, e a variância aumenta com o tempo.

O teste retorna 3 parâmetros principais:

1. **Estatística de teste:** Mede a distância entre os dados observados e a hipótese nula.
2. **Valor-p (p -value):** Indica a probabilidade de obter uma estatística de teste tão ou mais extrema que a observada, assumindo que a hipótese nula é verdadeira.
3. **Valores críticos:** Estabelecem os limites para determinar se a estatística de teste é suficiente para rejeitar a hipótese nula, em diferentes níveis de significância (1%, 5% e 10%).

Quanto menor for o p -value (mais baixo que os níveis de significância), mais forte é a evidência para rejeitar a hipótese nula, ou seja, a série é estacionária. Quanto à estatística de teste, quanto mais negativa ela for, mais forte é a evidência de que se rejeita a hipótese nula. Quanto aos resultados obtidos para cada uma das séries temporais de cada cliente, apenas uma delas foi dada como série não estacionária: a série da qualidade do ar do Cliente 1. Esta anomalia pode ser atribuída a dois fatores principais. Primeiramente, o sensor responsável pela recolha desses dados apresentou problemas de calibração persistentes, mesmo após a substituição pelos mesmos motivos. Em segundo lugar, como será detalhado na subsecção

3.2.2, os dados referentes à qualidade do ar no cliente 1 não apresentam um valor médio constante, dado que apresenta uma tendência crescente. Contudo, apesar da não-estacionariedade, os modelos usados nas simulações são capazes de captar padrões nos dados na mesma. No entanto, para o caso do cliente 1, não é de facto a situação ideal. Não foram aplicadas nenhuma transformação aos dados da qualidade do ar, reconhecendo que esta escolha pode ter impacto negativo nas simulações. Esta decisão foi tomada para manter a integridade dos dados originais, mesmo ciente das possíveis limitações que isso pode impor às análises subsequentes.

2.2.3 FL em IoT

Neste subcapítulo será abordado o FL num ambiente de IoT. Na primeira subsecção será dada uma pequena introdução e definição do conceito de IoT. Na segunda subsecção será estudada a integração do FL em sistemas com dispositivos IoT. Nas duas subsecções seguintes serão dadas as vantagens e desafios referentes a estes sistemas.

2.2.3.1 IoT

O IoT representa uma revolução na área da tecnologia da informação. Este conceito, derivado da combinação das palavras "Internet" e "*Things*", envolve a interligação global de redes de computadores e objetos físicos. O conceito "*Things*" abrange uma vasta gama de entidades tangíveis, desde dispositivos eletrónicos até objetos do quotidiano. A integração da Internet com objetos físicos, sejam vivos ou inanimados, altera a forma como interagimos com o mundo, permitindo a partilha de informação e recursos, dando origem a aplicações inovadoras [42].

Em [42], é definido IoT como “Uma rede aberta e abrangente de objetos inteligentes que têm a capacidade de se auto-organizar, partilhar informações, dados e recursos, reagindo e atuando face a situações e mudanças no ambiente”. Notar que não existe uma definição única de IoT, dado que existem diversas definições na comunidade científica.

2.2.3.2 Integração de FL em IoT

Em sistemas de IoT convencionais, o processamento de dados é feito de forma centralizada, deparando-se com limitações na gestão de um número crescente de dispositivos IoT distribuídos. A necessidade de redes robustas para transmitir dados entre dispositivos e centros de dados torna-se uma barreira à eficácia operacional [30].

Neste contexto, surge o FL. Ao descentralizar o treino de modelos e utilizar as capacidades computacionais dos próprios dispositivos IoT, o FL supera os desafios de escalabilidade, oferecendo uma solução eficiente para a gestão distribuída de um elevado número de dispositivos. Essa abordagem distribuída não apenas otimiza o processamento de dados localmente, mas também responde de forma eficaz à necessidade crescente de adaptabilidade e eficiência nos ambientes complexos da IoT [30].

No funcionamento de um sistema de FL-IoT, a distribuição e treino do modelo ocorre de forma descentralizada, proporcionando eficiência e preservando a privacidade dos dados. O processo inicia com a criação centralizada de um modelo global de ML, que é fragmentado e distribuído para os dispositivos IoT.

Cada dispositivo IoT, por sua vez, realiza o treino do seu modelo local com os seus próprios dados. Este treino local permite que o modelo aprenda padrões específicos aos dados do dispositivo, personalizando-o de acordo com as características locais.

Periodicamente, ou após um número definido de iterações de treino local, os modelos treinados localmente enviam atualizações para um servidor central. Estas atualizações são agregadas no servidor central, aperfeiçoando o modelo global. Este processo iterativo denomina-se por *model aggregation*, que consiste em agregar os modelos locais dos dispositivos IoT num único modelo global. Importante destacar que esta abordagem possibilita a atualização do modelo global sem a necessidade de aceder diretamente os dados brutos dos dispositivos, garantindo assim a privacidade e segurança dos dados [43]. Este processo iterativo permite que o modelo global melhore ao longo do tempo, uma vez que incorpora aprendizagens colaborativas provenientes de diversos dispositivos IoT. Na figura 11, é exemplificado um sistema típico de FL-IoT.

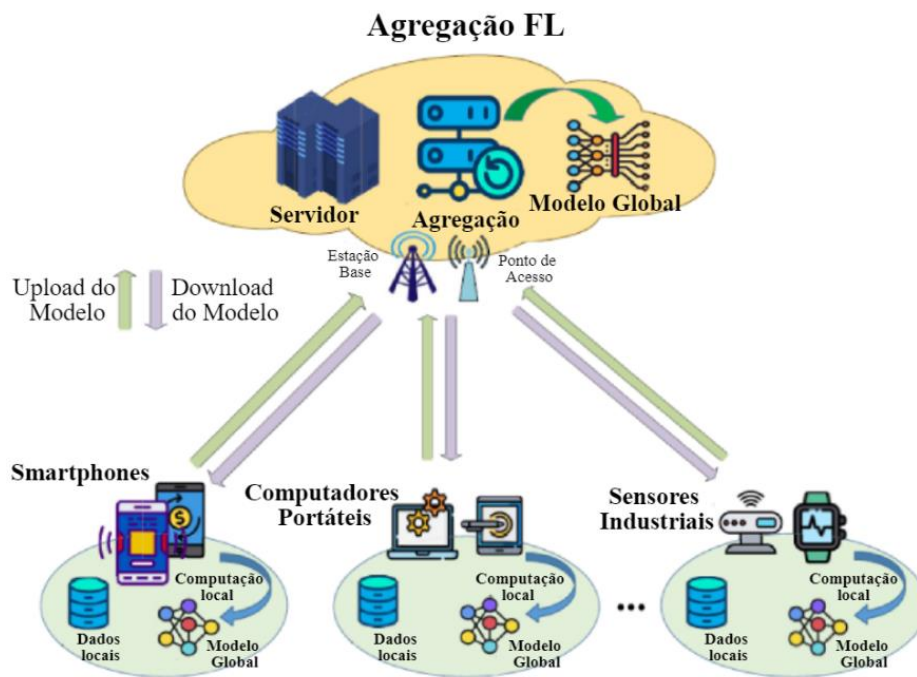


Figura 11 - Representação típica de um sistema FL-IoT (adaptada de [30])

2.2.3.3 Vantagens de sistemas FL-IoT

Privacidade dos dados – Como já foi mencionado anteriormente a abordagem ao FL proporciona a privacidade dos dados, dado que cada dispositivo de IoT existente no sistema não aprende nada mais do que a informação necessária ao seu funcionamento. Os dados brutos são mantidos localmente durante o processo de treino do modelo de agregação, sendo que apenas as atualizações do modelo são enviadas ao servidor central [44].

Latência reduzida na comunicação – Dado que não é necessário transferir os dados dos dispositivos IoT para o servidor, a abordagem ao FL reduz os atrasos na comunicação causadas pela *data offloading* [30]. O conceito *data offloading* refere-se a um método alternativo de transferência de dados nas horas de elevado tráfego de dados, por exemplo nas redes móveis, quando a rede se encontra sobrecarregada ou num caso de falha [45]. Ou seja, visto que não é necessária uma transmissão constante de dados para o servidor central, não será necessário utilizar outros canais de comunicação, conservando assim os recursos da rede.

Desempenho da aprendizagem do modelo – Devido às próprias limitações dos dispositivos, um único dispositivo de IoT pode não ter dados suficientes para aprender nem para melhorar o desempenho de um modelo por si mesmo. Com o FL todos os dispositivos podem colaborar

no treino do modelo, de modo que cada dispositivo possa beneficiar da recolha de dados feita por outros dispositivos, sem ter acesso a qualquer tipo de informação sensível [44]. Para além disso, ao utilizar diversos dispositivos com diferentes conjuntos de dados, os modelos de FL podem alcançar melhores taxas de convergência de resultados [30].

Escalabilidade – A escalabilidade do FL em sistemas de IoT possibilita uma gestão eficiente de um grande número de dispositivos, ao otimizar a distribuição de tarefas através da utilização dos recursos locais, facilitando assim o crescimento da rede. Para além disso com o FL, não é necessário a transferência de dados brutos, o que, só por si, contribui para o aumento da escalabilidade do sistema, principalmente no que diz respeito aos custos da comunicação, especialmente em redes de largura de banda reduzida [44].

2.2.3.4 Desafios em sistemas FL-IoT

Os desafios destes sistemas são praticamente os mesmos dos que já foram mencionados anteriormente no capítulo dedicado aos desafios do FL, tais como heterogeneidade, segurança, privacidade, custo, entre outros. Contudo, quando aplicado especificamente a sistemas de IoT, surgem novos desafios, sendo estes principalmente ligados aos *edge devices*. Ou seja, para além dos desafios ligados ao FL, surgem novos desafios ligados às próprias limitações dos *edge devices*, desde limitação na largura de banda e memória até à sua eficiência. Notar que este tópico é crucial, uma vez que esta dissertação é baseada no trabalho com Raspberry Pi's. Compreender os limites dos equipamentos a usar é de extrema importância para o sucesso da dissertação. Com isto os novos desafios associados a estes sistemas são os seguintes:

Consumo de energia – A maioria dos *edge devices* funcionam com recursos energéticos limitados, como baterias. No decorrer do desenvolvimento do modelo global, diversas tarefas são executadas, como comunicação, processamento local e agregação de dados. A realização dessas tarefas requer a utilização da energia armazenada na bateria do dispositivo. Em dispositivos com restrições energéticas, o consumo associado a essas tarefas pode-se tornar um desafio, afetando a eficiência e durabilidade do dispositivo, podendo até resultar em dispositivos com um comportamento intermitente [46].

Limitação de largura de banda – Alguns *edge devices* funcionam com uma largura de banda limitada, o que resulta em limitações na transferência de dados. No desenvolvimento do modelo global, várias tarefas essenciais são realizadas, incluindo comunicação, transferência de dados e agregação de informações. A execução eficiente dessas tarefas depende da largura de banda disponível. Em dispositivos com limitações na largura de banda, a transmissão e receção de dados pode ser um desafio, visto que pode resultar em transmissões mais lentas e

atrasos na resposta do sistema, comprometendo diretamente a eficiência do modelo e o desempenho dos *edge devices* [46].

Capacidade de processamento e Memória – Frequentemente *edge devices* apresentam limitações na capacidade de processamento e na quantidade de memória disponível. Tarefas exigentes em termos de processamento, como o treino de modelos, pode resultar em atrasos diminuindo o desempenho do mesmo. A falta de memória pode resultar em *overflow*, ou seja, o dispositivo tem falta de memória, então começa a escrever em posições de memória que já estão a ser usadas, corrompendo a memória do dispositivo. O que resulta na diminuição do desempenho do dispositivo [46].

As características intrínsecas dos *edge devices* têm impacto direto no desempenho local e global do sistema. Assegurar a disponibilidade constante num ambiente totalmente conectado e fiável representa um desafio complexo, agravado pela diversidade e mobilidade dos dispositivos. A variabilidade nos recursos exige abordagens adaptativas para garantir eficiência operacional. Estratégias como algoritmos de otimização de comunicação e técnicas de compressão de dados são essenciais para superar estes desafios, assegurando uma conectividade estável e um desempenho otimizado.

2.2.4 Algoritmos de agregação

Federated Averaging (FedAvg)

O FedAvg é um algoritmo de ML utilizado em cenários distribuídos. Foi desenvolvido para lidar com situações em que os dados de treino se encontram distribuídos entre diferentes dispositivos ou clientes, e a privacidade ou a descentralização dos dados é uma consideração importante. No processo do FedAvg, o servidor desempenha um papel crucial ao distribuir um modelo global para os clientes participantes. Cada cliente, por sua vez, coordena localmente a otimização do seu modelo com os seus próprios dados, recorrendo normalmente ao algoritmo de otimização SGD (Stochastic Gradient Descent) para aprimorar os parâmetros do seu modelo [46]. De seguida os modelos locais são enviados para o servidor, que efetua a agregação dos parâmetros otimizados, normalmente através de uma média ponderada (*averaging*), originando um novo modelo global [20]. Em [20], este algoritmo é caracterizado por 5 parâmetros:

- Fração de Clientes ou Participantes (**C**): Define a fração de clientes que participam ativamente em cada ciclo de comunicação.
- Tamanho do *minibatch* local (**B**): Indica o tamanho dos subconjuntos de dados processados por cada dispositivo durante a otimização local.
- Taxa de Aprendizagem (**η**): Regula os pesos dos parâmetros. Durante o treino de um modelo, o algoritmo de otimização, SGD neste caso, ajusta os pesos com base no gradiente da função de perda em relação a esses pesos. A taxa de aprendizagem é multiplicada nesse gradiente, determinando como devem ser ajustados os pesos.
- Número de *epochs* locais (**E**): Número total de iterações locais em cada cliente antes da atualização do modelo global.
- Decaimento da Taxa de Aprendizagem (**λ**): Ajusta a taxa de aprendizagem ao longo do tempo, contribuindo para uma aprendizagem mais precisa e estável.

FedProx

O algoritmo FedProx aborda de forma eficaz os desafios associados à heterogeneidade em redes federadas. Semelhante ao FedAvg em aspetos como a seleção de clientes em cada ciclo de comunicação e a agregação (média ponderada) de atualizações locais num modelo global, o FedProx apresenta algumas diferenças [47]. Em primeiro lugar, reconhece que os dispositivos na rede podem ter diferentes restrições de recursos, como hardware e níveis de bateria. Assim, o FedProx adapta-se, permitindo uma quantidade variável de trabalho conforme os recursos disponíveis em cada dispositivo. Adicionalmente, introduz um termo proximal para mitigar o impacto das atualizações locais, prevenindo possíveis divergências devido à heterogeneidade dos dados. Este termo proximal não só promove a segurança na incorporação de trabalho variável local, mas também limita a influência dessas atualizações, contribuindo para a estabilidade do algoritmo.

FedOpt

O FedOpt surge como uma solução inovadora para abordar a falta de adaptabilidade e flexibilidade no contexto do FL. Com semelhanças ao já estabelecido FedAvg, o FedOpt baseia-se na otimização abrangente na *framework* do modelo, introduzindo novos elementos para melhorar a convergência do modelo global [48]. Na sua essência, a abordagem do FedOpt envolve uma estrutura na qual os clientes realizam múltiplos *epochs* recorrendo a optimizadores, como ADAGRAD, ADAM e YOGI, para minimizar a função de perda nos seus dados locais. Este processo proporciona uma considerável flexibilidade aos clientes, permitindo adaptações locais mais eficazes. O servidor, por sua vez, desempenha um papel crucial na atualização do modelo global. Ao aplicar optimizadores baseados em gradientes na média das

atualizações dos modelos dos clientes, o FedOpt visa melhorar a convergência global do modelo. Notar que neste algoritmo tanto os clientes como o servidor são utilizados como otimizadores, adicionando uma dimensão colaborativa e descentralizada ao processo de otimização.

O algoritmo de agregação utilizado no trabalho desenvolvido no âmbito desta dissertação é o FedAvg porque é eficiente na agregação de modelos treinados localmente, reduzindo a comunicação entre dispositivos e lidando bem com dados distribuídos e heterogêneos.

2.2.5 Algoritmos

Nesta secção, serão apresentados os diversos algoritmos a serem avaliados para definir o modelo final a ser implementado. Um algoritmo é um conjunto de regras ou procedimentos para resolver um problema ou realizar uma tarefa. No contexto de *Machine Learning*, o algoritmo define como ocorre a aprendizagem e como são feitas as suas previsões. Um modelo, por outro lado é o resultado da aplicação de um algoritmo a um conjunto de dados específico. Assim, o modelo representa a aprendizagem obtida a partir dos padrões identificados nos dados. Enquanto, os algoritmos de agregação mencionados anteriormente, como o FedAvg, não atuam diretamente sobre os dados, mas sim sobre os modelos. Estes combinam os resultados dos modelos treinados nos diferentes dispositivos, trabalhando com os parâmetros ou pesos dos modelos treinados localmente.

A seleção dos algoritmos inclui tanto abordagens de *Machine Learning*, como Extra-Trees, Gradient Boosting, Random Forest e xGBoost, quanto técnicas avançadas de *Deep Learning*, como LSTM, GRU e RNN. O Facebook Prophet também será considerado, por ser adequado para séries temporais. Tanto os algoritmos de regressão como o Prophet servirão de referência, para comparar os resultados e determinar o algoritmo final a ser usado. Cada um dos modelos será brevemente introduzido a seguir.

Facebook Prophet: Algoritmo desenvolvido pela equipa de pesquisa do Facebook, projetado especificamente para previsões de séries temporais (time series). É conhecido pela sua simplicidade de uso e precisão em cenários com dados sazonais e com tendências. É particularmente eficiente em lidar com dados com padrões sazonais e anomalias, ajustando automaticamente os feriados e outras datas importantes [49].

XGBoost: É uma implementação eficiente e otimizada do algoritmo de Gradient Boosting. É conhecido pela sua velocidade e desempenho, especialmente em conjuntos de dados extensos.

Este algoritmo utiliza uma combinação de regularizações e otimizações computacionais para produzir modelos robustos e precisos [50].

Random Forest: Um método de *ensemble learning* que constrói múltiplas árvores de decisão durante o treino e produz a média das previsões individuais das árvores para melhorar a precisão. As Random Forest são robustas contra overfitting e podem capturar relações complexas entre as variáveis. Elas são especialmente úteis quando há muitas variáveis de entrada e interações entre elas [51].

Gradient Boosting: Algoritmo que combina vários modelos fracos (*weak learners*), como árvores de decisão, de forma sequencial. Cada modelo corrige os erros dos anteriores, ajustando-se com base no gradiente da função da perda, originando um modelo final mais preciso e robusto [52].

ExtraTrees: Método de *ensemble learning* que constrói múltiplas árvores de decisão de forma aleatória, tanto nas escolhas das divisões como no uso dos dados. Esta aleatoriedade torna o modelo mais robusto e menos sujeito a *overfitting* [53].

RNN (Recurrent Neural Networks): As RNNs são uma classe de redes neurais em que a saída de um passo anterior é utilizada como entrada para o passo seguinte. Isto é crucial em tarefas que exigem a previsão de sequências, como a próxima palavra numa frase, onde as entradas anteriores influenciam a previsão atual. Utilizam os mesmos parâmetros para cada entrada, o que reduz a complexidade dos parâmetros quando comparada com outros tipos de redes neurais. No entanto, uma das limitações deste modelo é a sua memória a curto prazo, o que pode prejudicar o seu desempenho em termos de retenção de informação quando aplicado a sequências longas [54].

LSTM (Long Short-Term Memory): Os modelos LSTM são uma versão melhorada das RNNs, que resolve o problema de dependências de longo prazo introduzindo um sistema de células de memória. A principal inovação das LSTM é a utilização de portas (gates) para controlar o fluxo de informações para dentro e fora da célula de memória, garantindo que as informações relevantes sejam mantidas por longos períodos e que as informações irrelevantes sejam descartadas. Esta arquitetura permite que as LSTM capturem dependências de longo prazo em dados sequenciais, tornando-as adequadas para tarefas como tradução automática, reconhecimento de voz e previsão de séries temporais [54].

GRU (Gated Recurrent Units): As GRU são uma versão simplificada das LSTM que também visa capturar dependências de longo prazo nos dados. Elas têm uma estrutura mais simples com menos parâmetros, o que pode resultar em tempos de treino mais rápidos enquanto ainda mantém a capacidade de lidar com dependências temporais. A ideia básica por trás das GRU

é usar mecanismos de *gating* para atualizar seletivamente o estado oculto da rede em cada passo de tempo. Estes mecanismos de gating são usados para controlar o fluxo de informações para dentro e fora da rede. As GRU possuem dois mecanismos de gating, chamados de reset gate (porta de reinicialização) e update gate (porta de atualização) [54].

2.3 Trabalho Relacionado

Em [55], foi apresentado um projeto que tem como objetivo demonstrar os benefícios do FL, com ênfase na privacidade e segurança. Nesse sentido foi simulado um cenário onde se pretende identificar se os trabalhadores de construção estão ou não a usar equipamento de proteção, neste caso o capacete. A solução proposta utiliza a ferramenta *Flower* para treinar um modelo descentralizado de classificação de imagem (MobileNetV2) num sistema de FL, envolvendo um *Raspberry Pi 4* e um computador como clientes. A finalidade deste projeto envolve o *Edge Impulse Python SDK*, que é utilizado para dar *deploy* do modelo diretamente, após a conclusão do treino do FL, em qualquer dispositivo que execute o código, incluindo o *RaspBerry Pi* e o computador, bem como o *deploy* do novo modelo global. A recolha de dados incluiu 583 imagens, divididas entre treino/teste para clientes, e pastas adicionais para teste no servidor central e modelo global. O treino foi realizado com 20 *epochs* e com um batch size 8, repetido seis vezes. Os resultados indicaram uma eficácia de 0.997 no modelo global. Após o *deploy* do modelo no *Edge Impulse*, foi realizada uma análise do desempenho do mesmo com um conjunto de imagens que não foram usadas no treino, obtendo uma eficácia de 93.75%. Em resumo, este projeto demonstra a eficácia do FL, obtendo modelos robustos sem transferir dados sensíveis para fora dos dispositivos locais. Destaca ainda o papel importante do *Edge Impulse* como maneira de dar *deploy* de modelos para qualquer dispositivo que execute o código.

Em [56], é destacada a potencial importância do FL em embedded devices, utilizando um cenário hipotético de classificação de imagem. Nesse contexto, considera-se a aplicação em frigoríficos inteligentes equipados com uma câmara capaz de detetar e classificar os objetos armazenados. Foi destacado que os utilizadores podem não querer partilhar esses dados e propõem o FL como solução para preservar a privacidade nesses dispositivos. Em seguida, destacaram o papel crucial da *framework* Flower, que apresenta um processo de instalação mais simplificado em comparação com TensorFlow e PyTorch. Outra vantagem

para a sua implementação é o facto de o único requerimento para o *embedded device* é o de possuir o Docker. Por último foi fornecida uma configuração exemplo com código.

Em [57], o projeto (EM^3) propõe uma abordagem integrada na recolha de dados, abstracção de informação (concentração de aspetos essenciais dos dados, ignorando características menos relevantes), recomendações e automações de forma a alcançar poupança de energia em ambientes domésticos ou profissionais. O sistema demonstra capacidade para lidar diariamente com milhares de eventos provenientes dos sensores, oferecendo análises personalizadas. A escalabilidade é evidenciada pela simulação de cenários distribuídos e processamento nos *edge nodes*, utilizando FL de forma a responder às necessidades dos utilizadores sem comprometer a privacidade. O sistema coordena sensores para recolher dados relevantes, analisados por módulos para fornecer informações essenciais a sistemas de automação e recomendação, utilizando o conceito de *micro-moments*. O objetivo principal deste estudo é recolher dados de implementações reais do sistema (EM^3) e do conhecimento adquirido da análise dos dados, de modo a descrever os detalhes no processamento de dados e informação, bem como simular a escalabilidade através de FL, com ênfase na viabilidade da implementação de LTSM e Federated LTSM em *edge devices*.

Em [58], foi desenvolvida e analisada uma *framework* FL destinada a treinar algoritmos de ML. A abordagem usada envolve a colaboração federada de clientes de FL com características heterogéneas, em que o objetivo central consiste em avaliar a eficácia do FL em ambientes com dispositivos heterogéneos que apresentam recursos limitados. O problema em questão foi problema típico de classificação de imagem com o propósito de desenvolver uma DNN com recurso ao FL. O sistema FL adotado segue uma arquitetura cliente-servidor, onde o servidor é representado por um computador com Windows e uma série de dispositivos Raspberry Pi que atuam como clientes de FL. Além disso, foi considerado também um conjunto de clientes virtuais para construir uma *framework* de FL que tenha em conta a heterogeneidade dos clientes. A conexão é estabelecida através de Wi-Fi e o algoritmo de agregação adotado foi o *FedAvg*. Foi utilizado o conjunto de dados CIFAR-10, recorrendo à linguagem de programação Python (plataforma Anaconda). Além disso a biblioteca PyTorch foi usada para treinar o modelo de NN, enquanto a biblioteca Flower, especializada em FL, automatiza as interações entre cliente e servidor de maneira eficiente. Foram encontrados diversos desafios associados a diferentes recursos computacionais, distribuição desigual de conjuntos de dados e problemas de aquecimento em dispositivos sem fio. As conclusões deste estudo proporcionam insights valiosos sobre mecanismos de arrefecimento, estratégias de compensação em redes heterogéneas, redução do *overfitting*, análise de escalabilidade e a integração bem-sucedida de *pretraining* e *Transfer Learning* em cenários de FL.

SOLUÇÃO PROPOSTA

Neste capítulo, será descrito o hardware usado para captura dos dados ambientais (dispositivos B-Health), incluindo a descrição dos seus componentes e a importância da integração num ambiente IoT. Será ainda abordada a estrutura e a implementação da solução desenvolvida. Para além de detalhar os componentes da solução, será abordada a *framework* utilizada para o *Federated Learning*, incluindo as razões pela qual foi escolhida e a sua importância para o projeto.

3.1 Dispositivo B-Health

Esta secção proporciona uma análise ao dispositivo B-Health, quanto aos seus constituintes e configuração. Desenvolvido pelo Uninova – Instituto de Novas Tecnologias, no âmbito dos projetos de investigação Europeus Smart4Health [59] e ICU4COVID [60]. Este dispositivo foi concebido para garantir a monitorização e análise em tempo real, de dados vitais de pacientes em contexto hospitalar, mantendo a informação segura. Cada um destes dispositivos consiste numa caixa que integra um Raspberry Pi e vários sensores, proporcionando assim uma solução integrada para recolha, processamento e monitoramento de dados ambientais. Nas subsecções seguintes serão detalhados os constituintes da caixa B-Health, bem como a sua configuração. Além disso, será abordado o motivo da escolha do Raspberry Pi.

3.1.1 Elementos do dispositivo B-Health

O dispositivo B-Health, consiste num Raspberry Pi e vários sensores, total de 5, os quais podem ser categorizados em duas zonas distintas: exterior e interior, sendo que 4 são exteriores e 1 é interior. Para além disso a caixa apresenta ainda uma entrada micro-usb (alimentação

da caixa), entrada SSH para conexão à rede local e um par de ventoinhas com o objetivo de contrariar o aquecimento do RaspBerry Pi. As figuras 12 e 13 representam respectivamente duas perspectivas distintas da caixa, uma visão frontal e outra uma visão lateral da caixa.

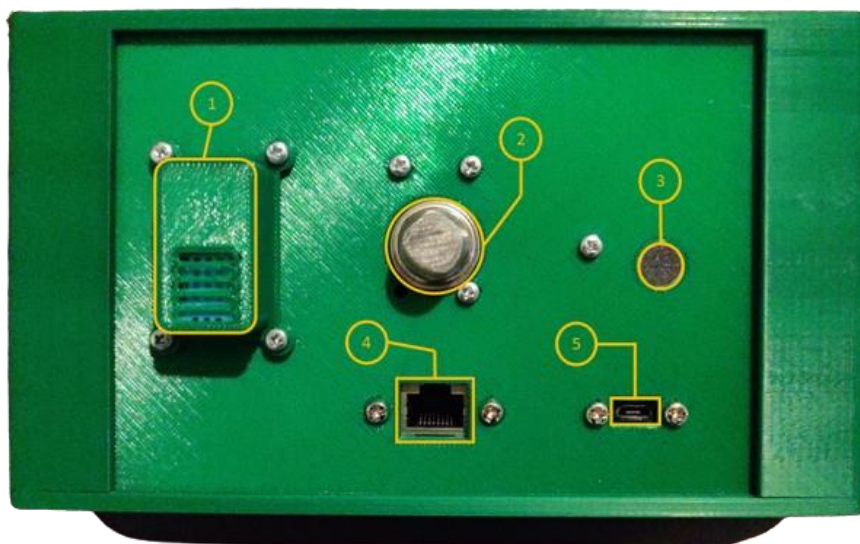


Figura 12 - Visualização frontal da caixa B-Health

Na figura 12, é possível identificar 5 elementos constituintes da caixa, referentes a 3 sensores exteriores, uma entrada SSH e uma entrada micro-usb. Os elementos, segundo a notação da imagem, são os seguintes:

- 1 – Sensor DHT11 – Temperatura e humidade
- 2 – Sensor MQ135 – Qualidade do ar
- 3 – Microfone – Ruído
- 4 – Entrada SSH – Ligação à rede local (através de um cabo de rede)
- 5 – Entrada micro-USB – Alimentação da caixa

No que diz respeito ao funcionamento da caixa, nota-se a presença de duas entradas, micro-USB e SSH. A entrada micro-USB é destinada à alimentação da caixa, enquanto a entrada SSH proporciona a conexão à rede local, utilizando um cabo de rede.

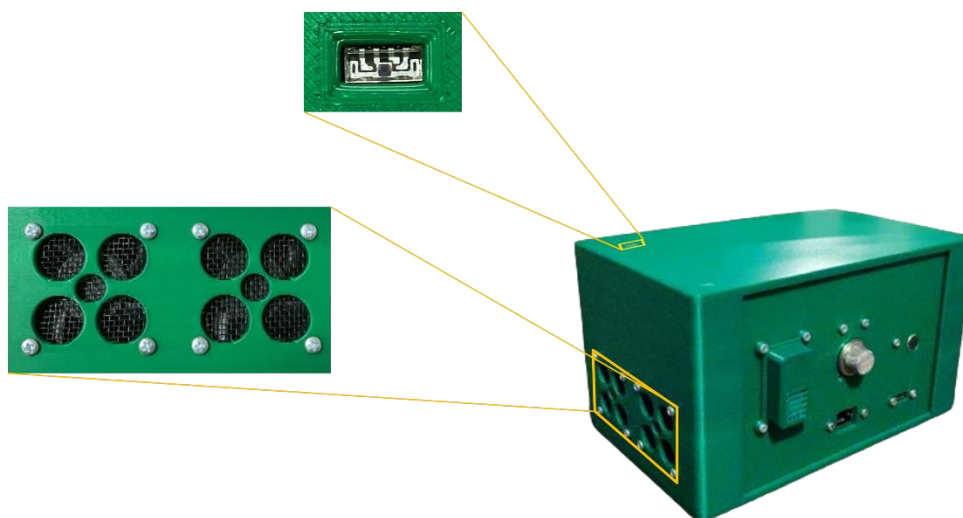


Figura 13 - Visualização lateral da caixa B-Health

Na figura 13 é possível observar o sensor exterior restante, bem como o par de ventoinhas usadas para contrariar o aquecimento do RaspBerry Pi. O sensor em questão é o VEML7700, responsável pela medição de dados relacionados com a luminosidade. Como já foi mencionado anteriormente, a caixa possui sensores exteriores e interiores. Todos os sensores exteriores já foram mencionados e podem ser observados nas figuras 12 e 13. Falta apenas mencionar o sensor interior. O sensor interior BMP280 mede a pressão atmosférica e temperatura no interior da caixa. Para além dos sensores e entradas da caixa, esta apresenta ainda um conversor A/D (Analógico para digital), cuja função é converter sinais analógicos em sinais digitais. Ou seja, transforma um sinal contínuo e variável no tempo num sinal discreto, que pode ser processado por um sistema digital, como um computador. Este conversor só é necessário para os sensores analógicos, ou seja, microfone (ruído) e MQ135 (qualidade do ar). Na tabela 1, é apresentado o nome de cada sensor para cada variável ambiental bem como as suas unidades.

Tabela 1 - Associação entre sensores, variáveis e as suas unidades

Nome do sensor	Variável Ambiental	Unidades
DHT11	Temperatura	$^{\circ}C$
BMP280	Temperatura	$^{\circ}C$
BMP280	Pressão Atmosférica	hPa
VEML7700	Iluminância	Lux
MQ135	Qualidade do Ar	Nível de CO_2 em ppm
Microfone	Ruído	dB

DHT11	Humidade	Humidade Relativa em %
-------	----------	------------------------

3.1.2 Configuração B-Health

Nesta subsecção será abordada a configuração da caixa B-Health, associada à ligação entre os sensores e a base de dados local. Notar que é de extrema importância entender todo este processo, de forma a facilitar a implementação de uma nova *framework* sobre esta ligação. Essencialmente nesta configuração, estão em operação seis scripts Python, dos quais três estão associados a sensores digitais (BMP280, DHT11 e VEML7700), um está associado ao conversor A/D (para os sensores analógicos de ruído e qualidade do ar), um está relacionado à base de dados, e outro está associado à interface. Todos os scripts de Python são automaticamente inicializados após a alimentação da caixa. De forma a facilitar a comunicação entre o Raspberry Pi e os vários sensores, recorre-se a um MQTT broker que surge como intermediário entre dispositivos, neste caso um Raspberry Pi e os sensores, que comunicam ao utilizar o protocolo MQTT. O objetivo principal do broker é garantir uma ligação eficiente entre dispositivos, em que neste caso em concreto se baseia no modelo *publish/subscribe*. Ou seja, os sensores existentes na caixa publicam (*publish*) dados em tópicos específicos do MQTT. O MQTT broker recebe os dados publicados e envia-os para qualquer dispositivo ou aplicação que tenha subscrito (*subscribe*) os tópicos de MQTT correspondentes. O script de Python da base de dados, subscreve os tópicos e consegue retirar os valores que estão na mensagem. Na figura 14, está representada a ligação entre a caixa (sensores e conversor) e o MQTT broker, onde é possível observar a publicação de informação em relação a cada sensor.

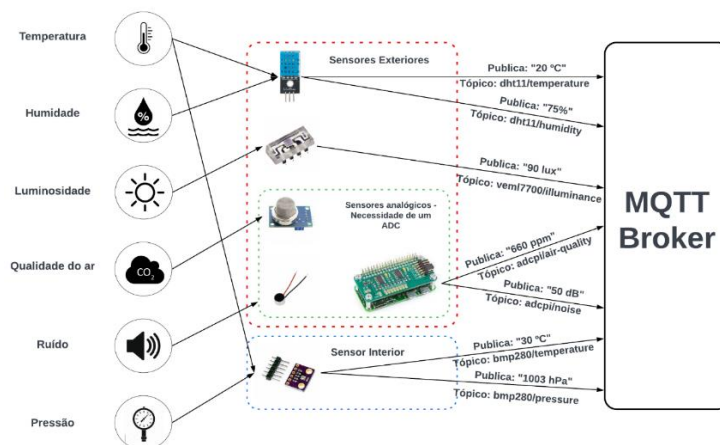


Figura 14 - Diagrama de ligação da parte do cliente, desde os sensores até ao MQTT broker

Quanto à interface, é possível visualizar os dados em tempo real através do URL `bhealthicuX:3000`, onde o "X" corresponde ao número atribuído à caixa. Este recurso proporciona uma monitorização contínua dos dados para uma análise em tempo real do ambiente. Na figura 15, é possível observar esta interface.

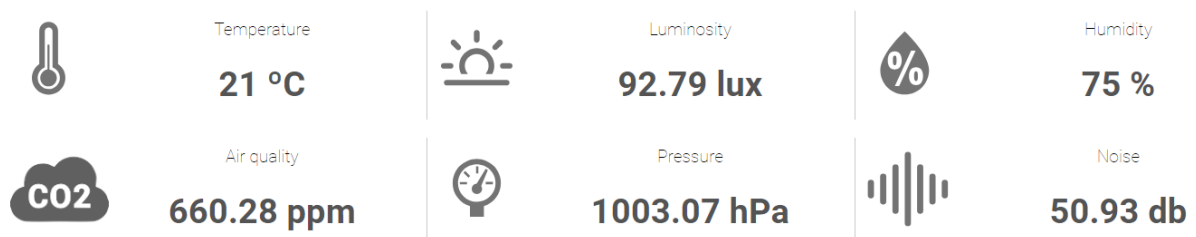


Figura 15 - Interface de monitorização em tempo real de uma caixa B-Health

Em relação à configuração do servidor, não há muito a dizer, somente que após o script Python subscrever os tópicos específicos do MQTT (script referente à base de dados), são extraídas informações relevantes das mensagens geradas pelo MQTT, como temperatura, humidade, luminosidade, entre outras. Estas informações são, então, armazenadas em diversas listas, sendo cada uma associada a um sensor específico. Após a conexão à base de dados MariaDB, a partir da função periódica `'send_data'` (função que se conecta periodicamente à base de dados), a função `'add_data'` é invocada, realizando, essencialmente, um comando INSERT para inserir dados nas tabelas correspondentes, nomeadamente (*Temperature*, *Luminosity*, *Humidity*, *Air Quality*, *Pressure* e *Noise*). Cada uma destas tabelas é composta por 4 colunas distintas: *timestamp*, *value*, *sensor* e *id*. O *timestamp* regista a data e hora da medição, o *value* representa o valor numérico obtido pelo sensor, o *sensor* identifica o sensor que efetuou a medição e o *id* serve para identificar de forma única cada registo. A tabela 2, apresenta a estrutura dos dados, especificando os tipos de dados associados a cada coluna.

Tabela 2 - Estrutura e tipo de dados de cada coluna referente a cada tabela dos sensores

Nome da coluna	Tipo de dados
Id	INTEGER
Timestamp	DATETIME
Value	FLOAT(8,4)
Sensor	CHAR(255)

Após a adição dos dados de todos os sensores na base de dados, é efetuado um *commit* para confirmar todas as alterações realizadas na base de dados. Por fim, a ligação é encerrada e todas as listas utilizadas para armazenar dados são limpas, evitando assim possíveis problemas quando estas acumularem mais dados. Na figura 16, encontra-se um diagrama a exemplificar o processo desde o MQTT Broker até à base de dados.

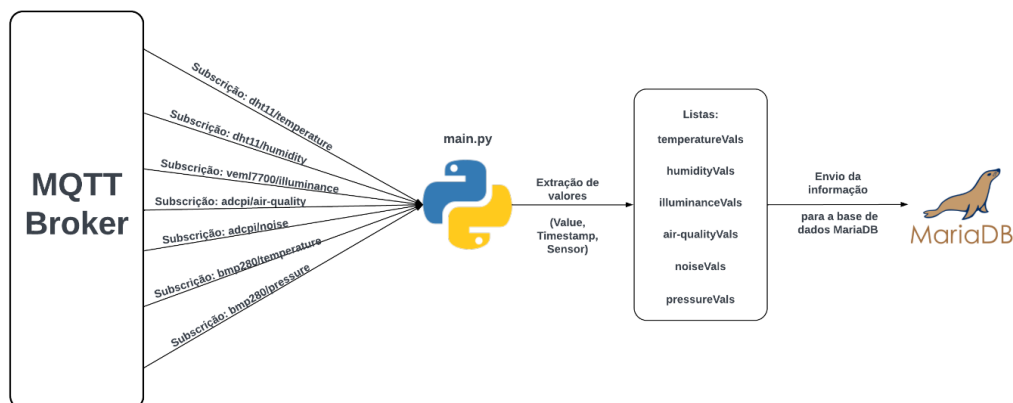


Figura 16 - Diagrama de ligação do lado do servidor, desde o MQTT Broker até à base de dados MariaDB

Para além da ligação das caixas B-Health à base de dados, é também possível aceder diretamente à base de dados através de uma ligação por SSH. Após ter acesso aos dados, é de extrema importância analisá-los de forma detalhada. Compreender o comportamento dos dados é crucial para identificar padrões, tendências e anomalias, permitindo uma análise mais precisa. Na próxima secção, serão realizadas análises detalhadas dos dados, incluindo a avaliação da disponibilidade, valores máximos, mínimos, média e mediana, bem como a realização de análises gráficas, como *boxplots* e variação temporal, com o objetivo de fornecer uma análise detalhada e crítica dos dados.

3.2 Solução desenvolvida

Nesta secção será apresentada a estrutura da solução desenvolvida, de forma a atingir os objetivos da dissertação. Inicialmente, a solução proposta para a recolha e processamento de dados em tempo real envolve a utilização de uma estrutura composta por:

- **3 Edge Devices (Caixas B-Health):** Responsáveis tanto pela recolha de dados quanto pelo treino de modelos de *Machine Learning*.
- **1 Servidor Central:** Destinado à coordenação das operações, armazenamento dos dados e gestão dos modelos treinados.

Durante a implementação desta estrutura inicial, surgiram vários desafios que comprometeram a eficiência e a robustez do sistema. Os edge devices, compostos por Raspberry Pi's 3, apesar de adequados para tarefas de baixa complexidade, mostraram-se insuficientes para lidar simultaneamente com a recolha de dados contínua e o treino de modelos de *Machine Learning* devido às seguintes limitações:

- **Capacidade de Memória Insuficiente:** Os dispositivos enfrentaram dificuldades em gerir a memória necessária para o processamento intensivo de dados e treino de modelos.
- **Poder de Computação Limitado:** O processador dos Raspberry Pi 3 não conseguiu atender às exigências computacionais elevadas, resultando num fraco desempenho.

Para abordar os problemas identificados, foi necessário recorrer a dispositivos com maior capacidade de processamento. Optou-se pela utilização dos Raspberry Pi's 4 devido às suas características superiores computacionais, como maior memória RAM, capacidade de armazenamento e desempenho de processamento. Na tabela 3, estão apresentadas algumas das especificações tanto do Raspberry 3 como do Raspberry 4.

Tabela 3 - Especificações dos Raspberry Pi's 3 e 4

	Raspberry Pi 4	Raspberry Pi 3
Modelo	Raspberry Pi 4 Model B	Raspberry Pi 3 Model B
RAM	4 GB	1 GB
CPU	Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz	Quad Core 1.2GHz Broadcom BCM2837 64bit
Arquitetura	aarch64 (ARM64)	aarch64 (ARM64)
Sistema Operativo	Debian GNU/Linux 12 (Bookworm)	Debian GNU/Linux 12 (Bookworm)

A estrutura inicial foi então reconfigurada para:

- **3 Raspberry Pi 3 (Caixas B-Health):** Designados exclusivamente para a recolha de dados. Estes dispositivos continuam a ser utilizados devido à sua adequação para tarefas de baixa complexidade.

- **3 Raspberry Pi 4:** Destinados ao treino de modelos de *Machine Learning*, aproveitando a sua maior capacidade de processamento e memória.
- **1 Servidor Central:** Mantém a função de coordenar todas as operações, tanto de agregação dos parâmetros dos modelos, bem como o treino dos mesmos.

Na figura 17, é ilustrada a nova estrutura da solução implementada.

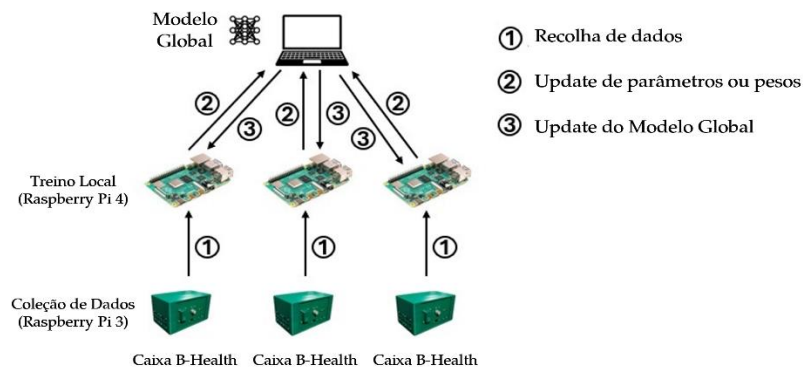


Figura 17 - Estrutura da solução implementada

Esta nova estrutura assegura uma separação clara das funções, otimizando tanto a recolha de dados quanto o processamento e treino de modelos, resultando num sistema robusto e mais eficiente. Na secção seguinte, será apresentada a *framework* Flower [61], que será utilizada para implementar o *Federated Learning*, explicando como esta contribui para um treino seguro de modelos de *Machine Learning*, preservando a privacidade dos dados dos utilizadores e melhorando o desempenho e precisão dos modelos treinados.

3.2.1 Flower

3.2.1.1 Vantagens e Desvantagens

O Flower é uma *framework* *opensource* especificamente desenvolvida para facilitar a implementação de sistemas de aprendizagem federada (*Federated Learning*) [61].

A escolha do Flower foi feita baseada em diversos fatores:

- **Treino Colaborativo:** Permite que múltiplos dispositivos colaborem no treino de um modelo comum, sem necessidade de partilhar dados sensíveis.
- **Gestão de Modelos:** Facilita a gestão de modelos de *Machine Learning*, desde a sua criação até à atualização contínua com novos dados e parâmetros.

- **Flexibilidade de Implementação:** Pode ser configurado para suportar uma ampla variedade de cenários e requisitos, adaptando-se facilmente a diferentes arquiteturas e fluxos de trabalho.
- **Monitorização e Melhoria Contínua:** Proporciona ferramentas e recursos para monitorizar o desempenho dos modelos e efetuar melhorias contínuas, garantindo que estes se mantenham atualizados e eficientes.
- **Facilidade de Implementação:** O Flower proporciona uma API simples e flexível que facilita a criação e gestão de sistemas de *Federated Learning*.
- **Suporte Multiplataforma:** A *framework* é compatível com várias plataformas e linguagens de programação, tornando-a uma escolha versátil para integrar com diferentes componentes da estrutura apresentada, incluindo os Raspberry Pi e o servidor central.
- **Comunidade e Documentação:** Flower possui uma comunidade ativa e uma documentação abrangente, oferecendo suporte contínuo e recursos para resolver problemas e implementar novas funcionalidades.
- **Segurança e Privacidade:** A utilização do Flower, garante que os dados sensíveis dos utilizadores nunca saem dos dispositivos locais, reduzindo o risco de violações de privacidade e conformidade com regulamentações de proteção de dados, como o GDPR.

Contudo o Flower, apesar de ser uma ferramenta robusta para o FL, apresenta algumas limitações, principalmente na escalabilidade e implementação de modelos em produção, dado que a escalabilidade não é gerida nativamente pelo Flower [62]. Quanto à privacidade esta oferece alguns mecanismos, como o *Server-side Clipping* e o *Client-side Clipping* com privacidade diferencial, mas ainda apresenta algumas limitações, dado que é aconselhável pelos desenvolvedores do Flower, o uso de outras ferramentas de forma a garantir a privacidade durante o treino local dos modelos nos clientes, como o Opacus e TensorFlow Privacy [63]. Apesar de tudo, neste contexto de investigação, o Flower proporciona uma base sólida para explorar e testar o FL. De seguida, será apresentado o método de funcionamento do Flower.

3.2.1.2 Método de funcionamento

O Flower permite a colaboração entre vários clientes para treinar modelos de ML sem a necessidade de centralizar os dados. O seu funcionamento pode ser descrito nos seguintes passos:

1. **Definição do Modelo e Estratégia:** O processo inicia-se com a definição do modelo de ML e da estratégia de FL. O modelo utilizado pode ser de qualquer tipo, desde que seja suportado pela *framework* Flower, como as redes neuronais (LSTM, GRU, RNN) e

modelos de regressão, desde que sejam utilizadas as bibliotecas necessárias. Para os modelos de DL, são importadas as classes LSTM, Dense, Dropout, SimpleRNN e GRU da biblioteca **Keras**. Para os modelos regressão, é utilizado o **scikit-learn**, principalmente as sub-bibliotecas **sklearn.ensemble** e **sklearn.linear_model**. Nas simulações feitas, os modelos de DL usados são do tipo Sequential, compostos por várias camadas, incluindo a camada **Dense**, que conecta todos os neurónios da camada anterior à seguinte, e **Dropout**, que desativa aleatoriamente neurónios durante o treino para evitar *overfitting*. Dos modelos apresentados na secção 2.6, apenas o Facebook Prophet não é suportado pelo Flower.

2. **Inicialização do Servidor:** O servidor centraliza o processo de treino e coordena a comunicação entre os clientes. Para iniciar o treino, o servidor configura e inicializa o modelo global, define a estratégia de treino e o algoritmo de agregação, como FedAvg, além de determinar o número mínimo e máximo de clientes aceitáveis para cada iteração de treino, entre outros fatores.
3. **Distribuição do Modelo para os Clientes:** Cada cliente recebe o modelo global do servidor através da função **get_parameters** e realiza o treino local com os seus próprios dados. Os dados são obtidos através de um **mysql.connector**, que faz a ligação à base de dados dos dispositivos B-Health, e executa algumas *queries* para obter dados dos diferentes sensores (temperatura, qualidade do ar, ruído, luminosidade e humidade), consoante uma amostragem definida pela variável **interval_seconds**. Devido à limitação de poder computacional dos dispositivos B-Health, os dados são recolhidos e processados por Raspberry Pi's 4.
4. **Tratamento de Dados:** Após a recolha são aplicadas algumas transformações necessárias de forma a agrupar os dados e gerar tanto uma avaliação qualitativa (de “Mau” a “Excelente”), como uma avaliação numérica (de 0 a 4) da qualidade do ambiente. Em seguida, é obtida uma avaliação numérica geral da qualidade ambiental ao somar as avaliações de todos os sensores no seu intervalo de tempo respetivo. No fim das transformações tem-se então os dados de entrada (X) compostos pelos valores dos sensores e intervalos de tempo, e (y) os dados de saída (alvo) referentes à avaliação numérica da qualidade do ambiente. Para os modelos de DL, é usada a função **create_sequences**, que gera sequências de dados de entrada (X) e de saída (y) com base num comprimento de sequência (**seq_length**). Esta função é essencial pois transforma os dados de séries temporais num formato adequado para modelos de DL, dado que estes operam sobre

sequências, permitindo que o modelo capture as dependências temporais e faça previsões com base nas relações entre os valores passados e futuros. No caso de modelos de regressão, estes não necessitam da função `create_sequences`, dado que não consideram dependências temporais ou relações sequenciais. Ou seja, baseiam-se na premissa de que o valor da variável de saída (y) é uma função direta dos valores das variáveis de entrada, ou seja, tratam as variáveis de entrada (X) como se fossem independentes no tempo, assumindo que o valor de (y) depende apenas das variáveis de entrada no momento atual.

5. **Treino Local:** Antes de iniciar o treino, são aplicadas normalizações aos dados, com auxílio do `MinMaxScaler`, garantindo assim que os dados estão numa escala adequada. Para além disso o conjunto de dados é ainda dividido em diferentes subconjuntos de treino e teste usando a divisão por séries temporais, `TimeSeriesSplit`, que preserva a ordem temporal. Cada cliente treina o seu modelo localmente, utilizando o último subconjunto de treino, com o método `fit`. Durante o treino, é monitorizada a função da perda (dada pelo erro quadrático médio), usando os *callbacks* como `History` e `ModelCheckpoint`, que registam a perda no treino e na validação em cada *epoch*.
6. **Avaliação Local:** A função `evaluate` avalia o desempenho do modelo nos dados de teste locais (utilizando também o último subconjunto) e retorna o erro quadrático médio (MSE) e a perda associada ao modelo.
7. **Envio dos parâmetros para o servidor:** Após o treino local, o cliente usa o método `get_parameters` para extrair os parâmetros atualizados (pesos do modelo) e envia-os de volta ao servidor.
8. **Agregação dos modelos:** O servidor aplica a estratégia de agregação definida, neste caso FedAvg, ao combinar os parâmetros enviados pelos diferentes clientes para criar um modelo global.
9. **Iteração:** Este processo de treino e agregação é repetido por várias iterações, até que o treino seja concluído. Neste caso, o treino é finalizado após 3 iterações.

O código inclui ainda funcionalidades para previsão futura através da função `forecast`. Esta função permite ao cliente fazer previsões das próximas 6 horas, recorrendo à última sequência de dados (de tamanho `seq_length`). A previsão é realizada passo a passo: o modelo prevê o próximo valor com base na última sequência e, em seguida, atualiza a sequência com a previsão para gerar a próxima previsão. Este processo é repetido até que se atinja o total de 6 horas de previsão. Além das previsões, o código gera gráficos para visualizar a comparação entre os valores reais e os valores previstos, permitindo uma análise visual da precisão das

previsões. Também são gerados gráficos que mostram a evolução da perda durante o treino, proporcionando uma visão clara do desempenho do modelo ao longo dos *epochs*.

No capítulo seguinte, será feita a análise de dados recolhidos bem como da qualidade do ambiente.

ANÁLISE DE DADOS E DA QUALIDADE DO AMBIENTE

Neste capítulo, será feita a análise dos dados recolhidos pelos dispositivos, com foco na disponibilidade dos dados e na sua representação gráfica. A visualização temporal será utilizada para ilustrar as variações dos dados ao longo do tempo, enquanto *boxplots* serão utilizados para analisar a distribuição e a variabilidade dos dados. Por fim, será descrita a abordagem utilizada para avaliar a qualidade do ambiente, detalhando os métodos e critérios usados na avaliação.

4.1 Análise de Dados

Nesta secção será feita a análise aos dados registados na base de dados dos dispositivos B-Health, conforme descrito na subsecção 3.1.2. A análise incluirá a disponibilidade dos dados, as estatísticas descritivas (máximo, mínimo, média e mediana), estas análises são complementadas pela elaboração de gráficos, tipo caixa de bigodes (*boxplots*) e gráficos de variação temporal. Como cada dispositivo está posicionado em diferentes locais do edifício, a análise será efetuada de forma individualizada para garantir a precisão e a eficácia do processo analítico.

4.1.1 Disponibilidade

A disponibilidade de dados refere-se a uma medida de avaliação da capacidade de acesso aos dados recolhidos. Por exemplo, caso a taxa de recolha seja 1 segundo, a cada 10 segundos deve-se ter acesso a 10 valores, enquanto num dia ter-se-á acesso a 86400 valores. De forma a testar esta métrica de avaliação, foi avaliada a disponibilidade nas 3 diferentes caixas, bem como para cada sensor. Para o teste foram recolhidos dados de 2 meses e meio. A disponibilidade foi calculada através da divisão entre a soma dos valores recolhidos ao longo dos 2 meses e 20 dias, e a situação ideal de recolha de valores, ou seja, 1 recolha a cada 5 segundos o que equivale a 1398434 valores recolhidos nesse período. É importante notar que a taxa de recolha de cada caixa não é constante, sendo que a mais recorrente é a de 5 segundos,

daí o uso da taxa de recolha de 5 segundos como referência para a situação ideal. Na tabela 4, estão apresentados os resultados referentes à disponibilidade (%) de dados para cada sensor presente em cada caixa.

Tabela 4 - Disponibilidade de cada sensor referente a cada caixa B-Health

Sensor	Disponibilidade (%)		
	Caixa 1	Caixa 2	Caixa 3
Temperatura	93.102	93.327	93.754
Ruído	88.95	88.176	88.032
Qualidade Ar	88.942	88.491	88.02
Humidade	93.125	92.827	92.024
Iluminância	98.331	98.013	97.956
Pressão	96.776	95.932	95.571

Em análise à tabela 4, é possível observar que nenhuma das caixas atinge os 100% de disponibilidade. Isso ocorre porque, conforme mencionado anteriormente, existem variações nas taxas de recolha ao longo do tempo. Quando estas taxas são superiores à taxa de 5 segundos, existe uma menor recolha de valores no mesmo intervalo de tempo, o que implica uma diminuição da disponibilidade dos dados. Observou-se que todas as caixas apresentaram níveis de disponibilidade bastante semelhantes, indicando uma uniformidade nos padrões de recolha nas diferentes caixas. Segue-se a análise gráfica e estatística dos dados.

4.1.2 Análise gráfica e estatística dos dados

Para compreender o comportamento dos dados, será realizada uma análise preliminar que inclui análises gráficas, como visualizações temporais e *boxplots*, para observar a dispersão e possíveis *outliers*. Serão ainda determinados os valores máximos, mínimos, bem como a média e mediana. Esta análise será realizada individualmente para cada caixa, considerando que elas podem exibir comportamentos distintos devido às diferentes localizações. De seguida serão apresentadas as tabelas e gráficos, dos dados referentes a cada dispositivo.

4.1.2.1 Análise estatística

Nesta sub-subsecção serão apresentados os resultados estatísticos obtidos referentes a cada dispositivo. Nas tabelas 5, 6 e 7 são apresentados os valores máximos, mínimos, a média e a mediana, bem como algumas características dos *boxplots* para cada dispositivo e os seus sensores. As tabelas incluem ainda o valor do 1.º e do 3.º quartil, que correspondem aos limites

inferior e superior do retângulo do *boxplot*, respectivamente, a amplitude interquartil, que é a diferença entre o 3.º e o 1.º quartil, e, por fim, os limites dos bigodes (*whiskers*), que servem para indicar a variabilidade dos dados fora dos quartis e identificar possíveis *outliers*. Os bigodes estendem-se até aos pontos de dados mais distantes que não são considerados *outliers*, oferecendo uma visualização clara da dispersão dos dados. Mais à frente será possível observar esses gráficos dos *boxplots*.

Tabela 5 - Estatísticas dos dados referentes a cada sensor do dispositivo 1

	Humidade (%)	Qualidade do Ar (ppm)	Temperatura (°C)	Iluminância (Lux)	Ruído (dB)	Pressão (hPa)
Máximo	70	8934.3604	155	526.23	75.99	1012.35
Mínimo	22	27.46	21	0	16.8	996.03
Média	51.654	2829.135	25.885	96.524	49.609	1004.636
Mediana	51	2937.29	26	24.88	48.38	1004.64
Q1 (1º Quartil)	49	1313.54	25	0	42.95	1002.36
Q3 (3º Quartil)	55	4096.27	27	246.07	53.83	1006.95
Amplitude Inter-quartil	6	2782.73	2	246.07	10.88	4.59
Limite Superior	64	8141.07	29	526.23	70.14	1012.35
Limite Inferior	40	27.46	22	0	26.64	996.03

Tabela 6 - Estatísticas dos dados referentes a cada sensor do dispositivo 2

	Humidade (%)	Qualidade do Ar (ppm)	Temperatura (°C)	Iluminância (Lux)	Ruído (dB)	Pressão (hPa)
Máximo	60.0	1533.9	31.0	927.13	74.92	1012.14
Mínimo	27.0	375.1	23.0	0	9.06	995.78
Média	45.571	579.481	27.018	165.245	46.979	1004.445
Mediana	46	567.27	27	70.5	46.44	1004.5
Q1 (1º Quartil)	43	512.31	26	0.0	41.03	1002.15
Q3 (3º Quartil)	49	631.78	28	234.55	51.99	1006.79
Amplitude Inter-quartil	6	119.47	2	234.55	10.96	4.639
Limite Superior	68	810.76	31	586.14	68.43	1012.14
Limite Inferior	34	375.1	23	0	24.59	995.78

Tabela 7 - Estatísticas dos dados referentes a cada sensor do dispositivo 3

Humidade (%)	Qualidade do Ar (ppm)	Temperatura (°C)	Iluminância (Lux)	Ruído (dB)	Pressão (hPa)
--------------	-----------------------	------------------	-------------------	------------	---------------

Máximo	48	939.58	29	950.63	76.36	1012.61
Mínimo	26	244.02	23	0	12.77	996.41
Média	37.716	425.011	26.281	131.229	47.013	1004.771
Mediana	38	429.02	26	57.14	47.14	1004.71
Q1 (1º Quartil)	35	383.42	25	0.0	42.0	1002.5
Q3 (3º Quartil)	40	464.17	27	188.93	52.23	1007.09
Amplitude Inter-quartil	5	80.849	2	188.93	10.229	4.59
Limite Superior	47	262.22	29	472.32	67.57	1012.61
Limite Inferior	28	585.38	23	0	26.66	996.41

Ao analisar as tabelas 5, 6 e 7, observa-se uma semelhança geral entre os dados, com a exceção dos dados de qualidade do ar do dispositivo 1. É importante notar que estes dados foram recolhidos após a substituição de todos os sensores de qualidade do ar, uma vez que, anteriormente, nenhum dos dispositivos fornecia leituras confiáveis nesse aspeto. Por isso, o conjunto de dados é relativamente limitado. Apesar da troca dos sensores, o desempenho do dispositivo 1 piorou ao longo do tempo. Além disso, na tabela 5, o valor máximo da temperatura é anormalmente elevado, o que pode indicar a presença de um *outlier*. Segue-se a análise gráfica.

4.1.2.2 Análise Gráfica

Nesta sub-subsecção, serão apresentados os gráficos temporais e os *boxplots* referentes a cada sensor, proporcionando uma visão detalhada da evolução dos dados ao longo do tempo e da distribuição dos mesmos. A apresentação dos resultados será feita por sensor, e não por dispositivo, permitindo a comparação simultânea dos três dispositivos para cada sensor. Notar que a reta verde nos *boxplots* representa a mediana. Nas figuras 18, 19 e 20 estão ilustrados os gráficos da iluminância referente aos dispositivos 1, 2 e 3, respetivamente.

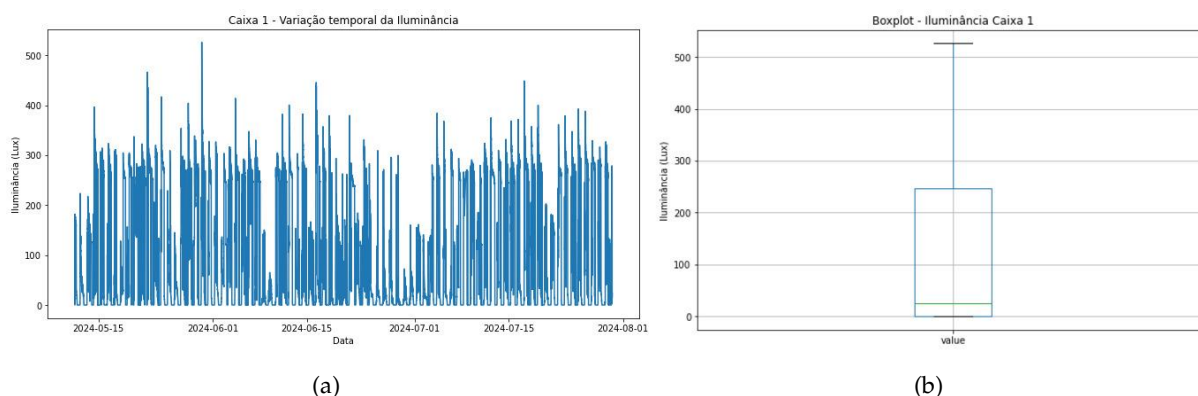


Figura 18 – Dispositivo 1 - Iluminância: (a) Variação temporal; (b) Boxplot

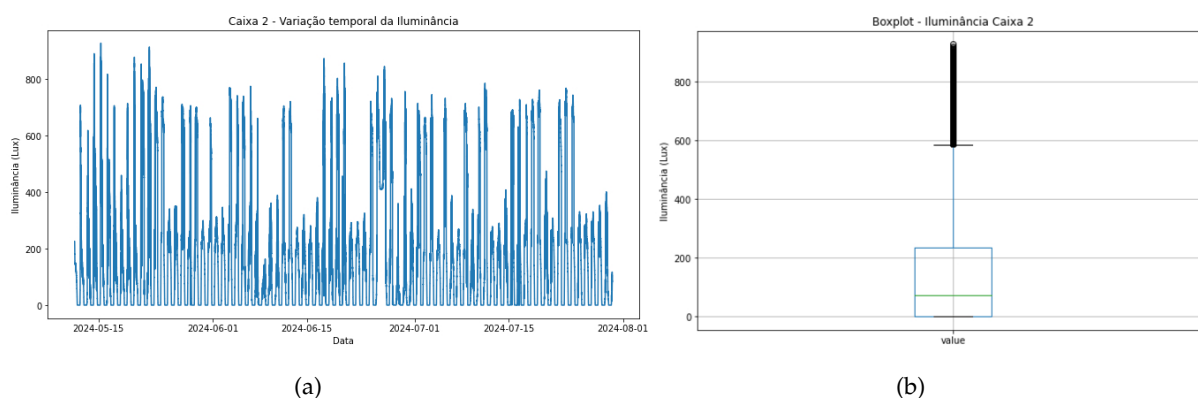


Figura 19 - Dispositivo 2 - Iluminância: (a) Variação temporal; (b) Boxplot

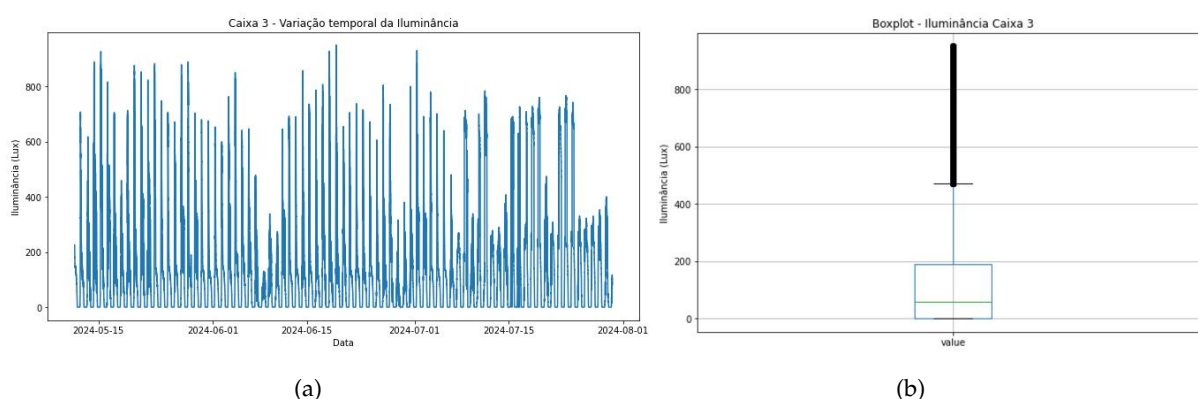


Figura 20 - Dispositivo 3 - Iluminância: (a) Variação temporal; (b) Boxplot

A análise às figuras 18, 19 e 20 revela, segundo os *boxplots*, a existência de *outliers*. No entanto, nos dispositivos 2 e 3, os *outliers* correspondem a valores superiores a 470 e 600 Lux, respetivamente. Estes valores são bastante legíveis correspondendo a uma boa iluminação, compatível com condições normais encontradas num escritório. O que sugere que os *outliers* identificados podem não ser irregulares. É ainda possível observar que os dados seguem um padrão consistente ao longo dos dias, sugerindo uma regularidade nas medições. Este

comportamento indica a presença de sazonalidade, evidenciando que as medições apresentam um padrão repetitivo ao longo do tempo. Segue-se a análise ao ruído.

Notar que, no caso do ruído, o gráfico da evolução temporal, figuras 21 (a), 22 (a) e 23 (a) não correspondem aos dados originalmente recolhidos, pois o gráficos apresentavam uma alta variância, o que torna impossível a análise do mesmo. De forma a diminuir a variância, foi usada uma amostragem de 15 minutos, ou seja, a cada intervalo de 15 minutos é feita uma média dos valores pertencentes a esse intervalo. Essa média corresponderá a um único ponto dos gráficos das figuras 21 (a), 22 (a) e 23 (a), de forma a suavizar os gráficos. Para os cálculos das tabelas 5, 6 e 7 foram usados os dados originais.

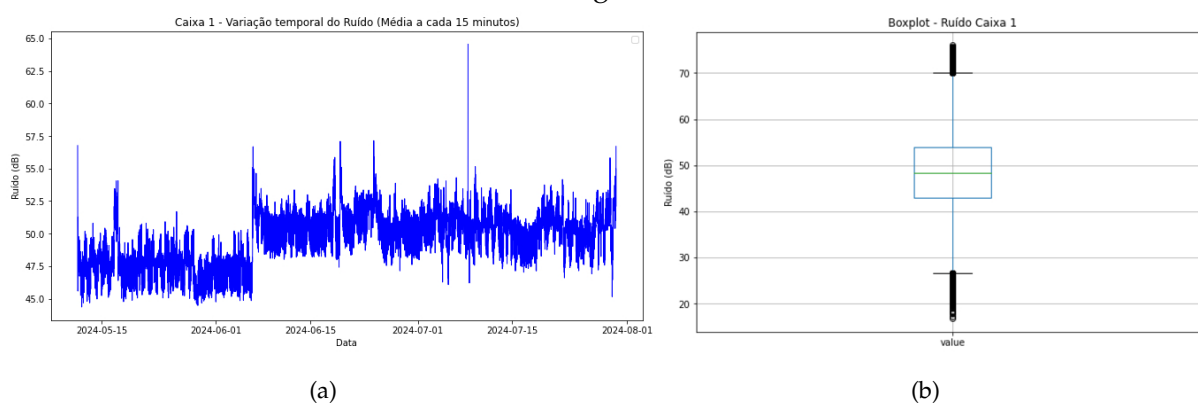


Figura 21 – Dispositivo 1 - Ruído: (a) Variação temporal; (b) Boxplot

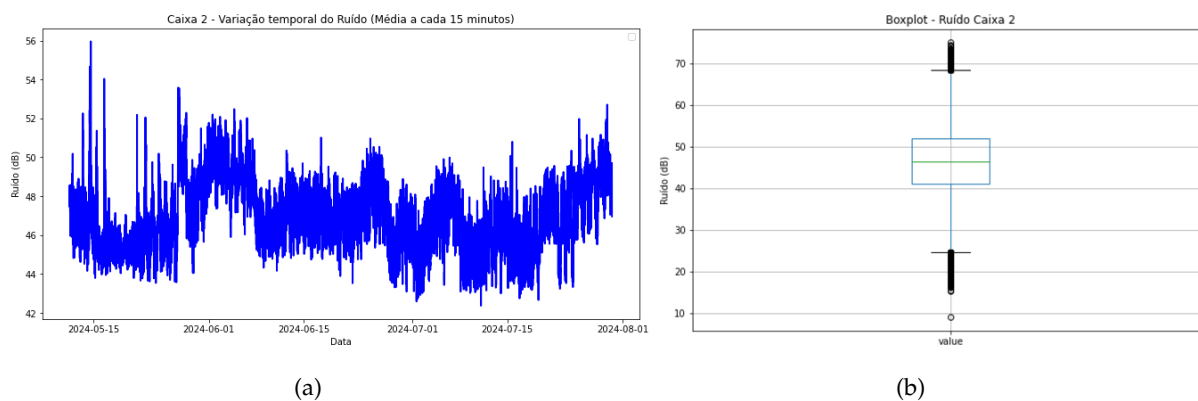


Figura 22 - Dispositivo 2 - Ruído: (a) Variação temporal; (b) Boxplot

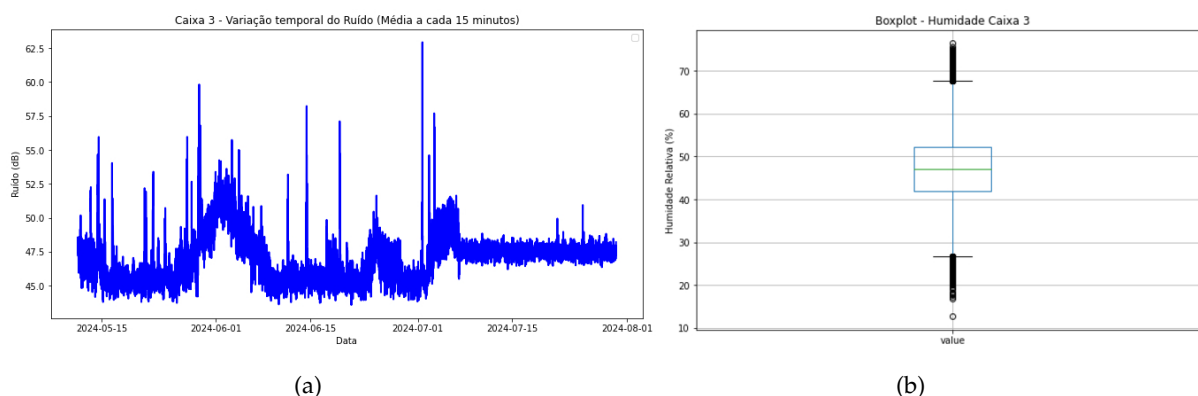


Figura 23 - Dispositivo 3 - Ruído: (a) Variação temporal; (b) Boxplot

Relativamente ao ruído, foram identificados alguns outliers, o que é esperado devido, principalmente, à sensibilidade dos sensores que podem captar sons intensos em determinados momentos e depois retornar a níveis normais. Além disso a localização dos dispositivos B-Health em áreas com movimento e atividade humana, como conversas, pode contribuir para essas variações. Relativamente à variação temporal dos dados, as conclusões são limitadas, uma vez que os gráficos das 21 (a), 22 (a) e 23 (a), não refletem exatamente os dados registados. No entanto, é possível afirmar que a média ronda os 50 dB, o que corresponde a níveis normais de ruído, em ambientes de escritório. De seguida, a análise gráfica da temperatura, figuras 24, 25 e 26.

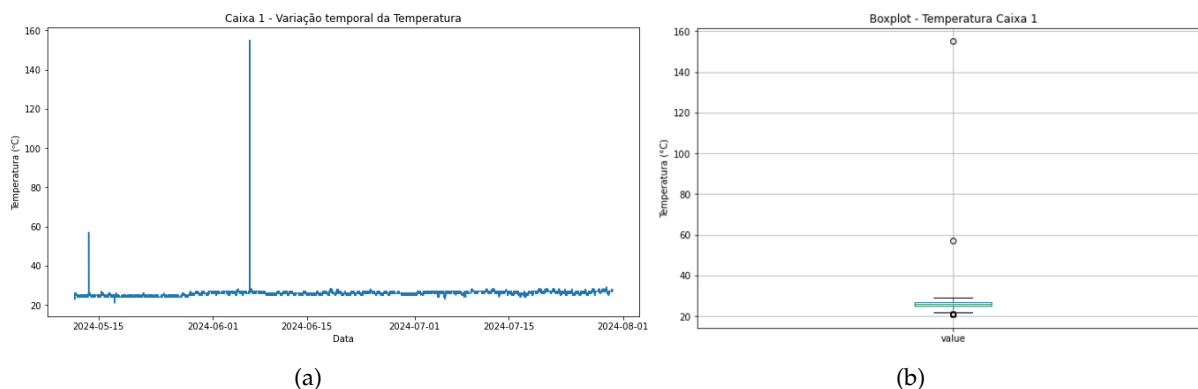


Figura 24 – Dispositivo 1 - Temperatura: (a) Variação temporal; (b) Boxplot

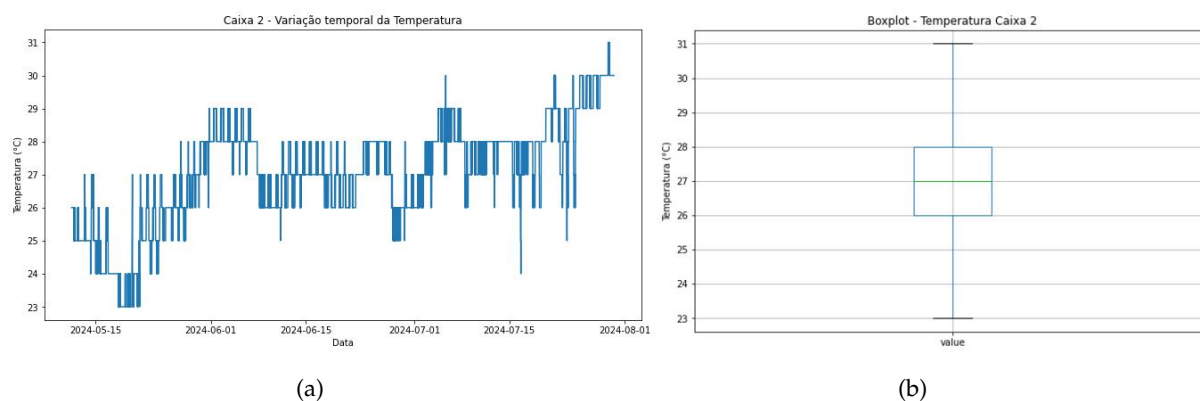


Figura 25 - Dispositivo 2 - Temperatura: (a) Variação temporal; (b) Boxplot

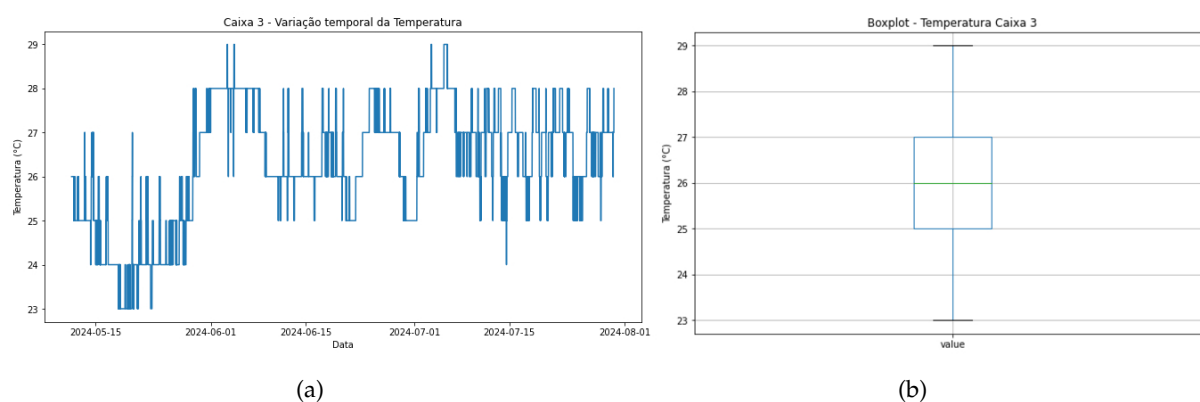


Figura 26 - Dispositivo 3 - Temperatura: (a) Variação temporal; (b) Boxplot

A temperatura no dispositivo 1 apresentou dois *outliers* significativos, figura 24, com valores de 155 °C e 68 °C, que são claramente anómalos e impossíveis na prática. Estes valores foram registados apenas num único registo e, em seguida, retomaram novamente para próximo dos 20 °C no registo seguinte. Estes valores discrepantes resultaram numa apresentação irregular tanto no gráfico da evolução temporal como no *boxplot*. Contudo, os dispositivos 2 e 3, apresentaram valores credíveis de temperatura, obtendo uma média de 27 °C e 26 °C, respetivamente. Quanto aos *outliers* do dispositivo 1, estes foram eliminados, sendo que para os intervalos de tempo afetados, foi calculada a média entre o valor da temperatura anterior e posterior ao intervalo de tempo do *outlier*. Segue-se a análise à qualidade do ar, figuras 27, 28 e 29.

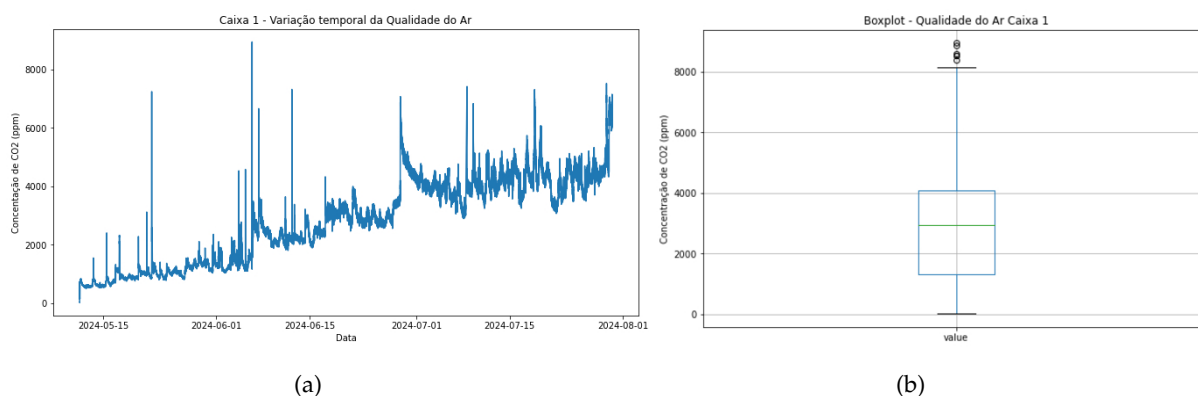


Figura 27 – Dispositivo 1 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot

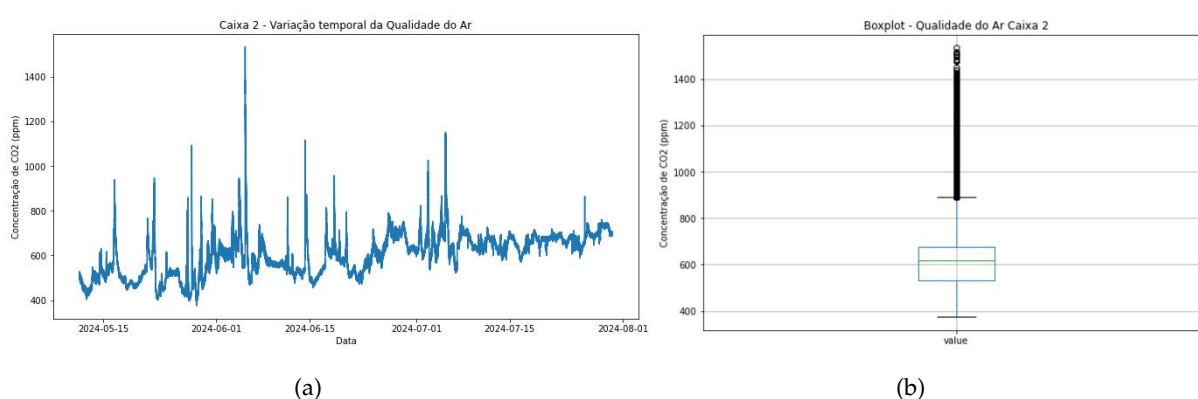


Figura 28 - Dispositivo 2 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot

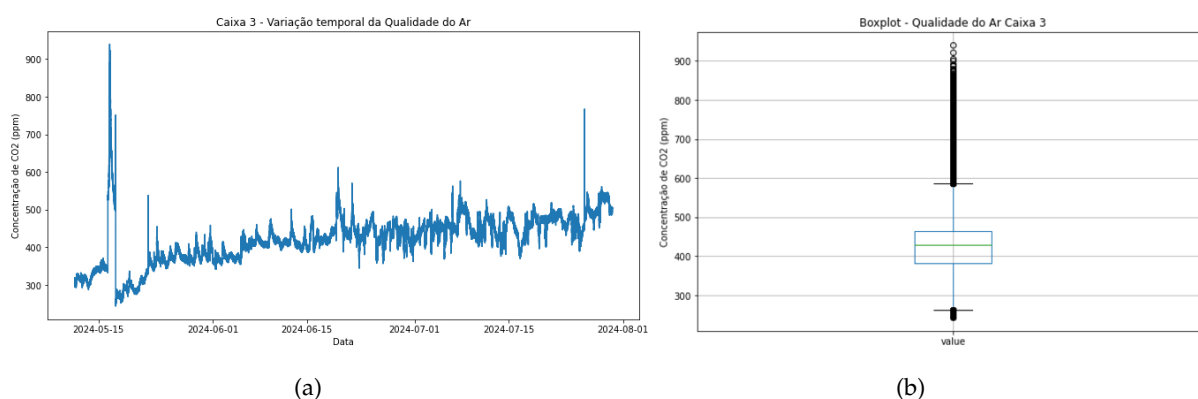


Figura 29 - Dispositivo 3 - Qualidade do Ar: (a) Variação Temporal; (b) Boxplot

Para a qualidade do ar, observou-se uma tendência crescente no dispositivo 1, o que pode sugerir um possível problema com o sensor, podendo este estar descalibrado. Dado que a partir de 2000 ppm, as condições ambientais da sala são bastante desfavoráveis. Como já foi mencionado estes dados foram recolhidos após a troca dos sensores da qualidade do ar em todos os dispositivos. Contudo, o sensor do dispositivo 1 deteriorou-se ao longo do tempo. Os dados recolhidos por estes sensores, serão os usados para as simulações, tendo em conta o dispositivo 1 terá uma deterioração da qualidade nos resultados das simulações. Quanto aos outros dois dispositivos, estes apresentaram alguns picos de concentração de CO₂, indicando

a presença de *outliers*, dado que a variação de CO₂ não é assim tão momentânea. Segue-se a análise da humidade, figuras 30, 31 e 32.

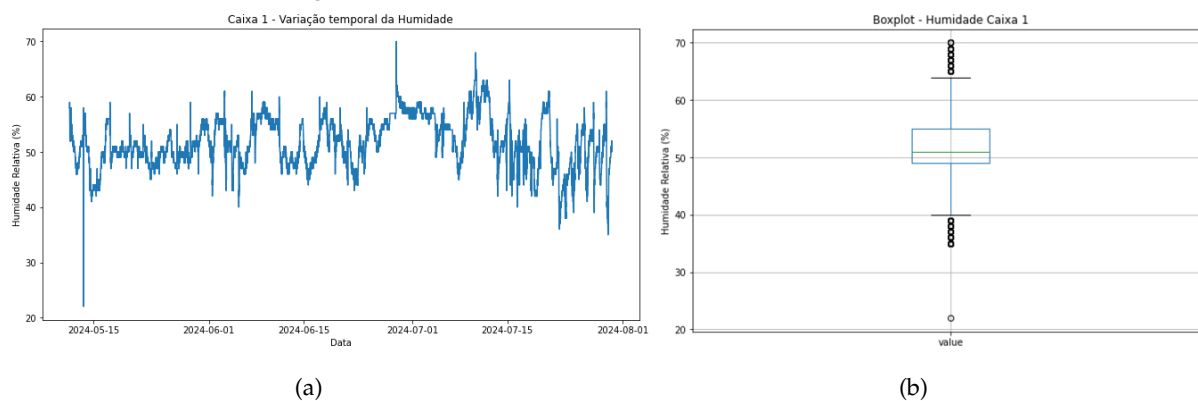


Figura 30 – Dispositivo 1 - Humidade: (a) Variação Temporal; (b) Boxplot

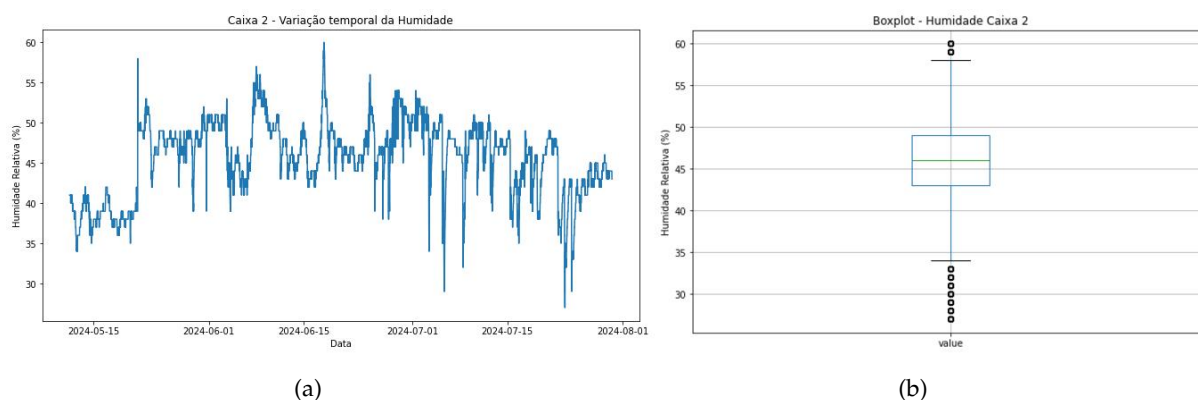


Figura 31 - Dispositivo 2 - Humidade: (a) Variação Temporal; (b) Boxplot

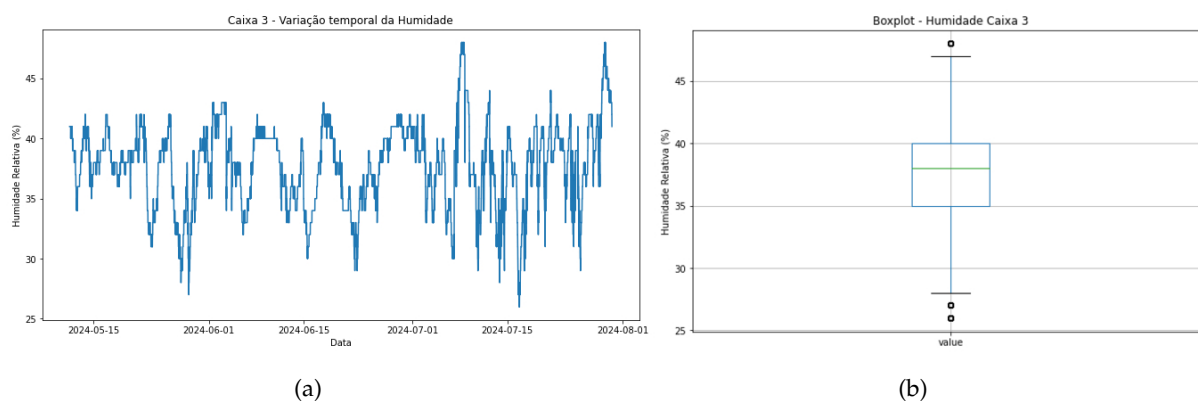


Figura 32 - Dispositivo 3 - Humidade: (a) Variação Temporal; (b) Boxplot

A análise da humidade revelou diversos valores que parecem ser *outliers*, devido às suas variações abruptas em intervalos de tempo consecutivos. Destacam-se alguns extremos que chegam até aos 22%, como é o caso no dispositivo 1, e outros casos que nem chegam aos

30%, no caso dos outros dois dispositivos. Estes valores chamam à atenção pois os valores medidos nos cinco segundos anteriores e posteriores andam na ordem dos 50%. Tanto o dispositivo 1 como o dispositivo 2 apresentaram uma média aproximada de 50 %, enquanto o dispositivo 3 apresentou uma média inferior, não ultrapassando os 40%. Por fim segue-se a análise à pressão atmosférica, figuras 33, 34 e 35.

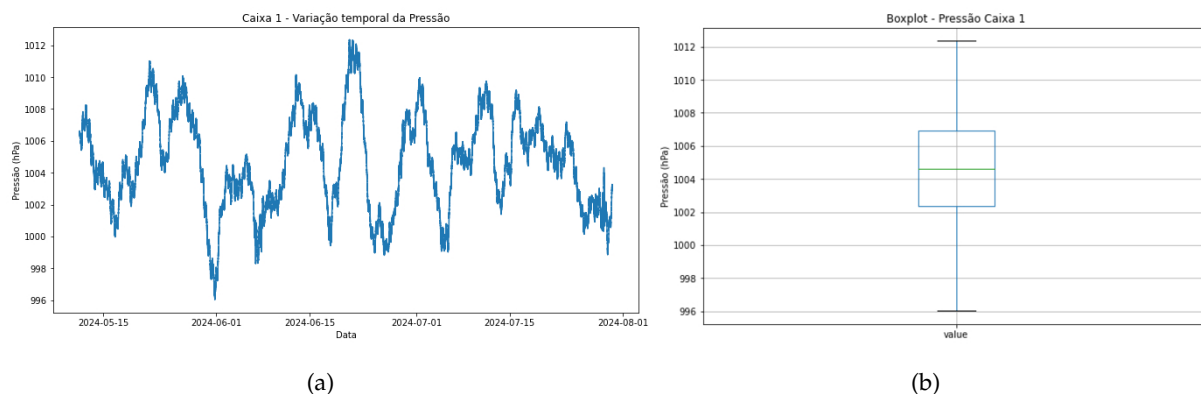


Figura 33 – Dispositivo 1 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot

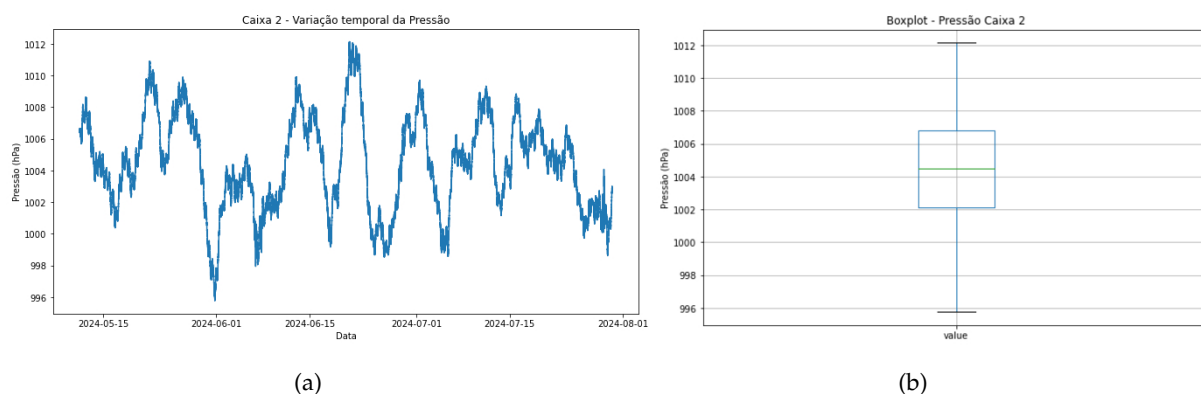


Figura 34 - Dispositivo 2 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot

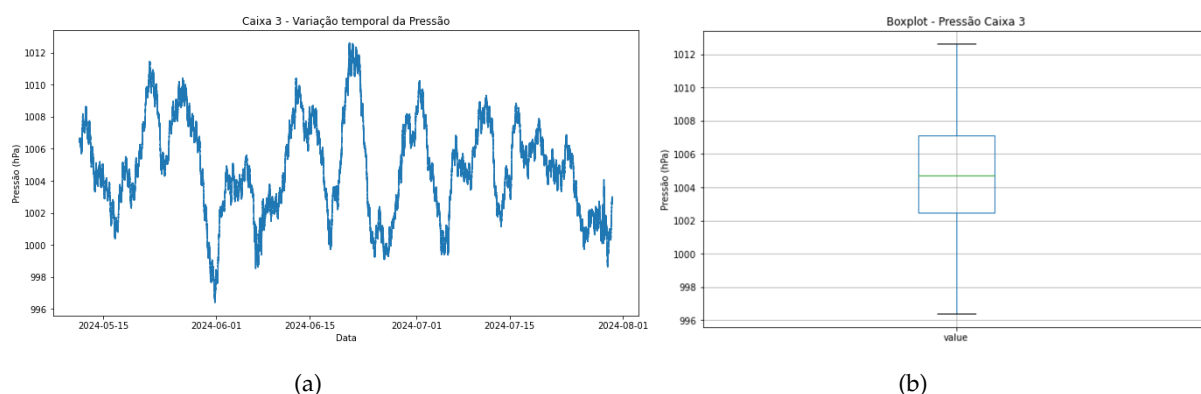


Figura 35 - Dispositivo 3 - Pressão atmosférica: (a) Variação Temporal; (b) Boxplot

Por fim, a pressão não apresentou *outliers* e mostrou pouca variação, o que é consistente com o esperado, dado que não existe muita variação na altitude das caixas em relação ao nível

do mar. Visto que a pressão apresenta uma variação mínima entre os dispositivos, e se mantém consistentemente próxima ao nível atmosférico normal de 1 atm (1010 hPa), a partir deste ponto, a pressão não será considerada para análise futura. Além disso, para a análise do ambiente das salas onde as caixas B-Health estão localizadas, a pressão não oferece informações significativas devido à sua estabilidade nas condições atuais.

4.2 Análise da qualidade do ambiente

De forma a analisar a qualidade do ambiente e devido à falta de uma classificação normas com os mesmos tipos de dados, recorreu-se a uma abordagem *naïve*, utilizando métodos simplificados e assumindo pressupostos básicos para facilitar a interpretação dos resultados. Para cada sensor foi definida uma escala qualitativa, que define numa nota de mau a excelente, cada valor do sensor dependendo na condição em que se encontra. Cada condição é formada por intervalo numérico em que caso o valor em estudo, esteja nesse intervalo é dada a nota que lhe pertence. Na tabela 8, estão ilustrados os intervalos de classificação considerados para cada um dos sensores:

Tabela 8 - Classificação para cada variável ambiental

Variável Ambiental	Classificação				
	Excelente	Bom	Suficiente	Fraco	Mau
Temperatura (°C)	20-25°C	19-20°C ou 25-26°C	18-19°C ou 26-27°C	17-18°C ou 27-28°C	Abaixo de 17°C ou acima de 28°C
Luminosidade (Lux)	500-750 lux	300-500 lux ou 750-1000 lux	200-300 lux ou 1000-1250 lux	100-200 lux ou 1250-1500 lux	Abaixo de 100 lux ou acima de 1500 lux
Humidade (%)	30-50%	25-30% ou 50-60%	20-25% ou 60-70%	15-20% ou 70-80%	Abaixo de 15% ou acima de 80%
Qualidade do Ar (ppm)	Abaixo de 800 ppm	800-1000 ppm	1000-1200 ppm	1200-1400 ppm	Acima de 1400 ppm
Ruído (dB)	Abaixo de 45 dB	45-50 dB	50-60 dB	60-70 dB	Acima de 70 dB

De forma a quantificar as notas de cada sensor foi também considerada uma cotação numérica dependendo da nota qualitativa de cada um dos sensores. Ou seja, dependendo da

qualificação entre Mau e Excelente, é dada uma condição numérica entre 0 e 4 respetivamente, como é possível observar na tabela 9.

Tabela 9 - Qualificação qualitativa e numérica da qualidade dos valores medidos pelos sensores

Avaliação Qualitativa	Avaliação Numérica
Mau	0
Fraco	1
Suficiente	2
Bom	3
Excelente	4

Para determinar a qualidade do ambiente é feita uma soma das avaliações numéricas referentes a cada um dos sensores. Por isso, a qualidade do ambiente é determinada através da soma de 5 valores, referentes à avaliação numérica de cada sensor, como é demonstrado na equação (1).

$$A_N = \sum_{i=1}^n A_{T_i} + A_{L_i} + A_{H_i} + A_{QA_i} + A_{R_i} \quad (1)$$

Onde A_N se refere à avaliação numérica da qualidade do ambiente, n ao número de intervalos de tempo, A_{T_i} à avaliação numérica da temperatura no intervalo de tempo i , A_{L_i} à avaliação numérica da luminosidade no intervalo de tempo i , A_{H_i} à avaliação numérica da humidade no intervalo de tempo i , A_{QA_i} à avaliação numérica da qualidade do ar no intervalo de tempo i , e A_{R_i} à avaliação numérica do ruído no intervalo de tempo i . A avaliação qualitativa é feita da mesma forma que foi no caso dos sensores, de mau a excelente, só que a escala numérica é diferente como é possível observar na tabela 10.

Tabela 10 - Intervalos numéricos da avaliação numérica tendo em conta a avaliação qualitativa

Avaliação Qualitativa	Avaliação Numérica Intervalo	
	Mínimo	Máximo
Mau	0	4
Fraco	5	9
Suficiente	10	14
Bom	15	19
Excelente	20	-

Como a análise é realizada através da soma das avaliações numéricas relativa a cada sensor, surge um novo problema derivado das diferentes taxas de recolha dos diversos sensores. Ou seja, numa situação ideal num dia ter-se-ia 17280 valores recolhidos por cada sensor, equivalente a uma taxa de recolha de 5 segundos. Nesta situação ao agrupar os diferentes

valores de cada sensor em diferentes colunas, ter-se-ia para cada intervalo de tempo espaçado de 5 segundos, valores referentes a cada sensor. No entanto, como é possível observar na tabela T1, referente à disponibilidade de cada sensor, isso não acontece. O que resulta em valores em branco em certos intervalos de tempo, o que torna a soma da avaliação da qualidade do ambiente impossível. A solução para este problema passa por agrupar valores que se encontram dentro de um certo intervalo, através de uma média ponderada desses valores. Por exemplo, supondo que se agrupa os valores dos vários sensores a cada 10 segundos, e dentro desse intervalo, o sensor da temperatura apresenta 2 valores, o valor final para esse intervalo seria a média desses 2 valores. Para determinar o intervalo de agrupamento, foi necessário calcular primeiro o intervalo máximo entre 2 intervalos de tempo consecutivos, de forma a garantir que não existem valores em branco e conclui-se que o intervalo máximo é de 12 segundos. Assim, optou-se por agrupar os dados a cada 12 segundos, resultando em 7200 valores diários para cada sensor. Na tabela 11 está representada uma pequena parcela da tabela dos valores agrupados.

Tabela 11 - Tabela agrupada com um único intervalo de tempo, espaçado de 12 segundos, para cada sensor

Intervalo de Tempo	Humidade	Iluminância	Temperatura	Qualidade do Ar	Ruído
2024-04-05 00:00:00	59	0	22	366.01001	31.46999
2024-04-05 00:00:12	59	0	22	365.64999	47.20000
2024-04-05 00:00:24	59	0	22	365.56500	39.89500
2024-04-05 00:00:36	59	0	22	365.82999	46.03499
2024-04-05 00:00:48	59	0	22	365.53667	47.44333
2024-04-05 00:01:00	59	0	22	365.91999	44.31001

ANÁLISE DE RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos com todos os modelos mencionados anteriormente na subsecção 2.2.5, como LSTM, GRU, RNN, Random Forest, Extra-Trees, xGBoost e Gradient Boosting. O desempenho de cada modelo será avaliado através da função de perda (loss), que mede a precisão das previsões com base nos dados de treino, calculada pela diferença entre os valores previstos pelo modelo e os valores reais. A duração das simulações também será um fator importante na avaliação dos modelos. A primeira secção explicará como as simulações serão conduzidas e quais parâmetros são mais relevantes. Nas subseqüentes, serão apresentados os resultados dos modelos de regressão tradicionais, que servirão de referência, bem como dos modelos de *Deep Learning*.

5.1 Definição de parâmetros

Todos os resultados aqui apresentados, têm por base a infraestrutura anteriormente descrita, composta por: 3 clientes e 1 servidor, e por 3 iterações de comunicação entre os clientes e o servidor. Uma iteração de comunicação, consiste num ciclo completo de comunicação entre o servidor e os clientes. Ou seja, numa fase inicial o servidor envia o modelo global para o cliente. O cliente recebe o modelo e realiza o treino localmente com os seus próprios dados. Após o treino, o cliente envia as atualizações do modelo para o servidor (como gradientes ou ajustamento de pesos dos parâmetros). Por fim o servidor agrega as atualizações dos clientes e distribui o modelo global atualizado para os clientes, concluindo assim a uma iteração ou ronda de comunicação. Além da natureza computacional dos clientes, existem certos parâmetros que podem ser alterados para além dos parâmetros de cada modelo como por exemplo:

- **Média por intervalo:** Calcula a média de um intervalo específico de valores. Este parâmetro é projetado para capturar a média dos valores contidos dentro de um intervalo

determinado. Por exemplo, se se especificar um intervalo de tempo, o parâmetro calculará a média dos valores incluídos nesse intervalo. Esta funcionalidade é útil para suavizar dados, ou seja, reduzir a variabilidade num conjunto de dados, ou para destacar tendências ao longo de um período específico, permitindo uma análise mais precisa e focada em determinadas partes dos dados.

- **Comprimento da sequência:** Este parâmetro define o número de valores passados que serão considerados na previsão dos valores futuros. Determina a extensão da sequência de dados históricos utilizada para realizar a previsão. Ajustar este parâmetro pode influenciar significativamente a precisão das previsões, permitindo ajustes com base na complexidade e na natureza dos dados em questão.
- **Horizonte de Previsão:** Este parâmetro determina o número de valores futuros que serão previstos pelo modelo. Por exemplo, se o tamanho da previsão for definido como 360, significa que o modelo irá prever os próximos 360 valores na sequência temporal ou na série de dados em questão. A escolha deste parâmetro é crucial para ajustar a precisão e a utilidade das previsões, adaptando-se às necessidades específicas do problema e às características dos dados.

Antes de apresentar os resultados obtidos de cada um dos modelos, serão exibidos na secção seguinte os valores reais correspondentes ao horizonte de previsão. Ou seja, dado que cada um dos modelos mencionados irá prever um período de 6 horas, seria fundamental apresentar os valores reais referentes a esse período para facilitar a análise comparativa entre os resultados das previsões e os dados reais observados. Notar que, uma vez que a previsão é feita com base na avaliação numérica da qualidade do ar, os dados reais foram igualmente transformados para representar uma avaliação numérica da qualidade do ambiente.

5.2 Dados reais relativos ao horizonte de previsão

Nesta secção serão apresentados os dados reais relativos ao horizonte de previsão. A figura 36 (a), (b) e (c), correspondem respetivamente aos dispositivos B-Health 1, 2 e 3.

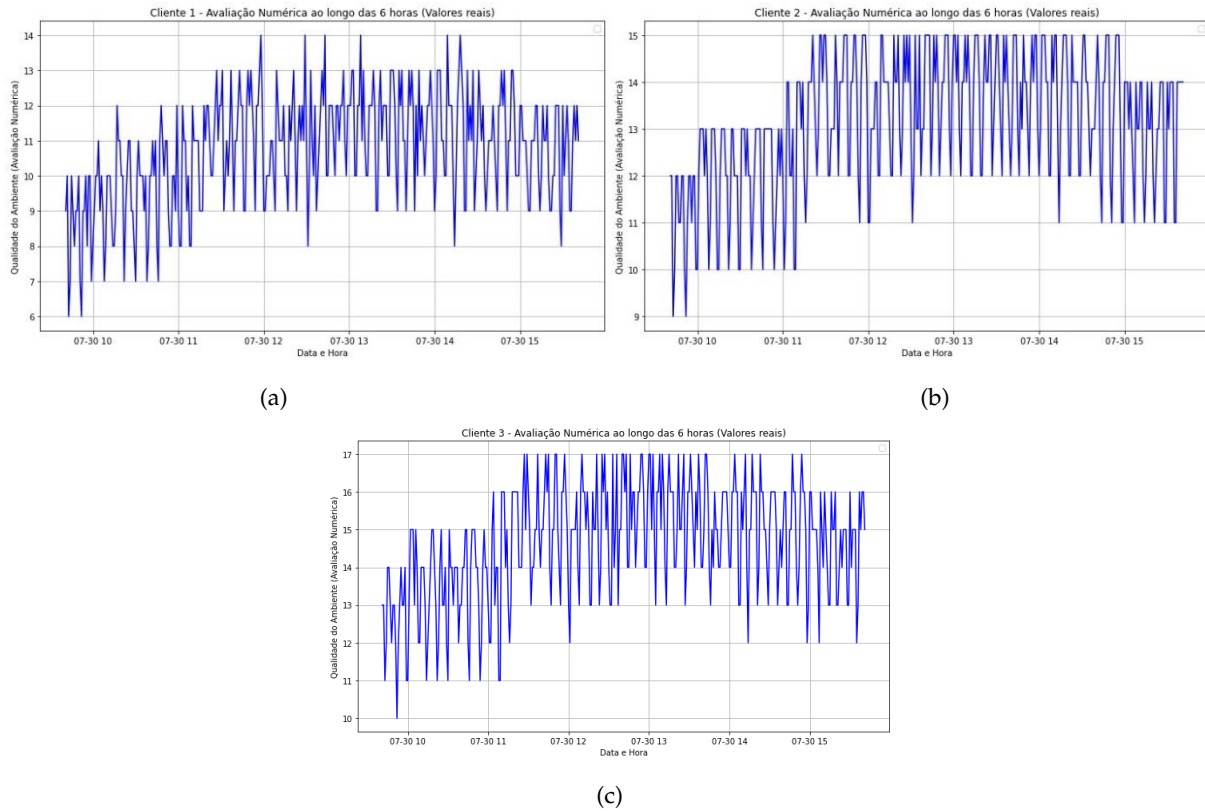


Figura 36 – Variação temporal da Qualidade do Ambiente (Avaliação Numérica) no horizonte de previsão para cada dispositivo: (a) Dispositivo 1; (b) Dispositivo 2; (c) Dispositivo 3

Em análise à figura 36, observa-se uma semelhança no comportamento da qualidade do ambiente em todos os 3 dispositivos. Na parte da manhã, todos apresentam níveis mais baixos da qualidade do ambiente, possivelmente devido à ausência de luminosidade e possivelmente outros fatores como o ruído. Após as 11 horas, a qualidade do ambiente parece aumentar até aproximadamente às 12 horas e em seguida mantém um comportamento relativamente constante, com ligeiras oscilações, acompanhadas por algumas descidas e subidas na qualidade do ambiente. Em termos médios, a sala onde se encontra o dispositivo B-Health 3 parece ser o que apresenta a melhor qualidade do ambiente, dado que apresenta a média mais alta entre os dispositivos atingindo uma média perto dos 15 valores, correspondendo a uma boa qualidade do ambiente (em média). O dispositivo 2 ficou por uma média de 13.5, enquanto o dispositivo 1 apresentou os valores mais baixos, devido a uma descalibração do sensor da qualidade do ar, o que resultou numa avaliação da qualidade do ambiente mais baixa em comparação com outros dispositivos. Ao analisar a figura, é ainda possível concluir que este método de avaliação é de facto muito simples, o que o torna muito instável por vezes. Idealmente uma escala de avaliação mais robusta seria o melhor.

Nas próximas secções, serão apresentados os resultados obtidos, tanto dos modelos de referência como os de *Deep Learning*. Os modelos de referência, como ExtraTrees, Random

Forest, xGBoost e Gradient Boosting, fornecem um ponto de partida, permitindo comparar e determinar se os modelos de *Deep Learning* atingem ou não melhores resultados. Serão ainda apresentados os resultados do modelo Facebook Prophet, modelo este que é próprio para séries temporais. Primeiro, serão exibidos os resultados dos modelos de referência com as suas diferentes amostragens, bem como do Prophet. Em seguida, serão apresentados os resultados dos modelos de *Deep Learning*, cuja eficácia depende de alguns parâmetros adicionais, os quais serão explicados posteriormente. As previsões obtidas dos modelos serão ainda comparadas com os valores reais obtidos, referentes à figura 36.

5.3 Modelos de referência

Como mencionado anteriormente, estes modelos são de extrema importância, pois fornecem uma base de resultados que permite comparar com os resultados obtidos, quanto à previsão, dos modelos de *Deep Learning*, bem como com os dados reais referentes ao horizonte de previsão. Nesta secção, serão apresentados os resultados dos modelos de referência, com as suas diferentes amostragens. A utilização de múltiplas amostragens não está necessariamente relacionada com os modelos discutidos nesta subsecção, mas sim com problemas específicos que serão abordados e explicados posteriormente. Todas as simulações foram realizadas com a *framework* Flower, exceto o modelo Prophet, que não é compatível com esta *framework*, sendo simulado apenas de forma independente. Ou seja, os dados foram carregados diretamente para o modelo e foram recolhidos os resultados, tanto da perda como da previsão obtida, sem a integração do Flower. Na subsecção seguinte serão apresentados os resultados referentes ao modelo Facebook Prophet. Posteriormente serão apresentados os resultados dos modelos de regressão.

5.3.1 Facebook Prophet

Como este modelo não é compatível com o Flower, as simulações foram realizadas sem recorrer ao *Federated Learning* (ou seja, sem a participação de clientes e servidor). Neste cenário, o Prophet foi configurado com regressores que representam variáveis como temperatura, humidade, qualidade do ar, ruído e iluminação. Além disso, foi aplicada uma técnica de validação (*cross-validation*). Esta técnica avalia o modelo ao segmentar os dados em várias partes, onde alguns segmentos são usados para treino e outros para teste. Notar que os

segmentos de teste são posteriores aos de treino, dado que se trata de séries temporais e não faria sentido testar em valores passados e treinar com valores futuros. Foi ainda realizada uma (*Grid-Search*) de forma a percorrer diversas iterações do modelo e otimizar o seu desempenho, ao identificar a combinação ideal de parâmetros do modelo Prophet. A avaliação do modelo Prophet foi realizada utilizando a função de perda MSE (Erro Quadrático Médio), que avalia a precisão das previsões do modelo. O MSE pode ser calculado através da equação (2).

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2)$$

Onde n é o número total de observações, y_i é o valor real da i -ésima observação e $\hat{y}_i$ é o valor previsto para a i -ésima observação. Na tabela 12, estão exibidos os resultados obtidos do Facebook Prophet.

Tabela 12 - Resultados obtidos pelo Facebook Prophet da função da perda para cada amostragem

Amostragem	MSE
1 min	0.3291
5 min	0.3218
10 min	0.3161
15 min	0.3097

Em análise à tabela 12, é possível concluir que os resultados obtidos da função da perda foram mais altos do que era esperado, visto que se trata de um modelo especializado para séries temporais. Contudo, notar que no caso do Prophet, as simulações foram executadas sem recorrer ao *Federated Learning* e ao Flower, o que pode ter uma influência negativa nos resultados do mesmo. Na figura 37, está ilustrada a previsão futura de 6 horas, referente ao dispositivo B-Health 1. Dado que a variação temporal é bastante semelhante em todas as amostragens testadas, foi selecionada apenas uma amostragem com a menor perda: 15 minutos.

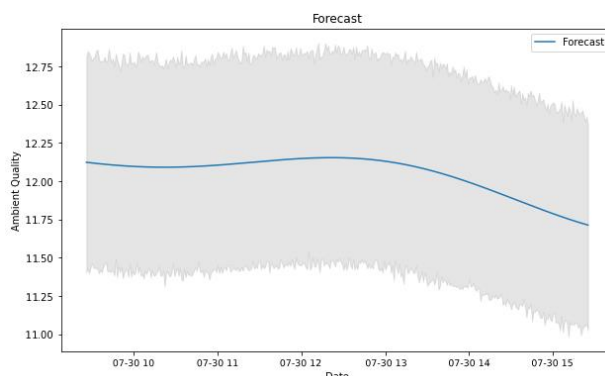


Figura 37 - Previsão do modelo Prophet referente a 6 horas para uma amostragem de 15 minutos

Em análise à figura 37, os valores para a amostragem de 15 minutos estão à volta de 12. Em comparação com os dados reais referentes ao horizonte de previsão do dispositivo B-

Health 1, figura 36 (a), é possível concluir que o modelo Prophet não se adaptou muito bem aos dados e à sua variação temporal. Dado que, para o caso do Prophet, a sua variação temporal permanece bastante constante ao longo do tempo, enquanto na realidade deveria haver uma variação mais acentuada ao longo da previsão. Sendo este um modelo próprio para séries temporais, os resultados obtidos ficam aquém do esperado. Contudo, este modelo foi aplicado de forma independente, sem recurso ao Flower e ao FL, o que pode justificar, em certa parte, os resultados obtidos.

5.3.2 Modelos de Regressão

Nesta subsecção serão apresentados os resultados obtidos nos modelos de regressão, nomeadamente ExtraTrees, Random Forest, xGBoost e Gradient Boosting. Cada simulação envolve 1 servidor e 3 clientes, com a execução de 3 iterações de treino. Para cada simulação, será apresentado o resultado obtidos para função da perda, dada pela equação (2). Na tabela 13, estão apresentados os resultados obtidos para cada um dos modelos. Por fim, serão apresentados os resultados gráficos referentes à previsão de cada modelo para cada amostragem. Para evitar que o documento se torne muito extenso em imagens, serão apenas apresentados os resultados do Cliente 1 para cada um dos modelos de regressão.

Tabela 13 - Resultados obtidos para os modelos de regressão para cada amostragem

Amostragem	Iterações Treino FL	MSE			
		Random Forest	ExtraTrees	Gradient Boosting	xGBoost
1 min	Iteração 1	5.70E-05	0.0001398	6.69E-05	7.73E-05
	Iteração 2	5.70E-05	0.00013742	6.59E-05	7.72E-05
	Iteração 3	5.77E-05	0.00013781	6.62E-05	7.72E-05
5 min	Iteração 1	0.000120392	0.00030958	0.000136048	0.000175975
	Iteração 2	0.000117227	0.00031579	0.000133469	0.000175913
	Iteração 3	0.000122461	0.0003217	0.000134602	0.000175485
10 min	Iteração 1	0.000196694	0.0004567	0.000230443	0.000203433
	Iteração 2	0.000193019	0.00046606	0.000230792	0.000203179
	Iteração 3	0.000192936	0.00046232	0.000228767	0.000203124
15 min	Iteração 1	0.000207155	0.00055799	0.000255819	0.000275061
	Iteração 2	0.000197632	0.00055148	0.000264184	0.00027498
	Iteração 3	0.000197261	0.00054754	0.00025849	0.000274713

Em análise às tabelas 12 e 13, é possível concluir que o modelo Prophet apresentou os piores resultados em termos de perda. Apesar de ser um modelo especializado na previsão de séries temporais, o Prophet foi avaliado sem a utilização do *Federated Learning*, e por se tratar

de um modelo de séries temporais, possivelmente necessitaria de mais dados para melhorar o seu desempenho. O facto de ter um conjunto de dados composto apenas por 2 meses e meio, pode ter sido um fator, dado que o Prophet é bom em detetar padrões de sazonalidade, mas para este caso pode não ter sido suficiente. Quanto aos restantes modelos de regressão, estes apresentaram resultados bastante melhores em relação à perda. Além dos baixos valores da perda, as previsões resultantes dos modelos de regressão, foram bastante positivas quando comparadas com os valores reais da figura 36. De seguida serão apresentados os resultados obtidos graficamente, quanto à previsão futura de 6 horas. Para cada modelo, serão apresentados os resultados gráficos para cada uma das amostragens. As figuras 38 a 41, representam respetivamente Random Forest, ExtraTrees, xGBoost e Gradient Boosting.

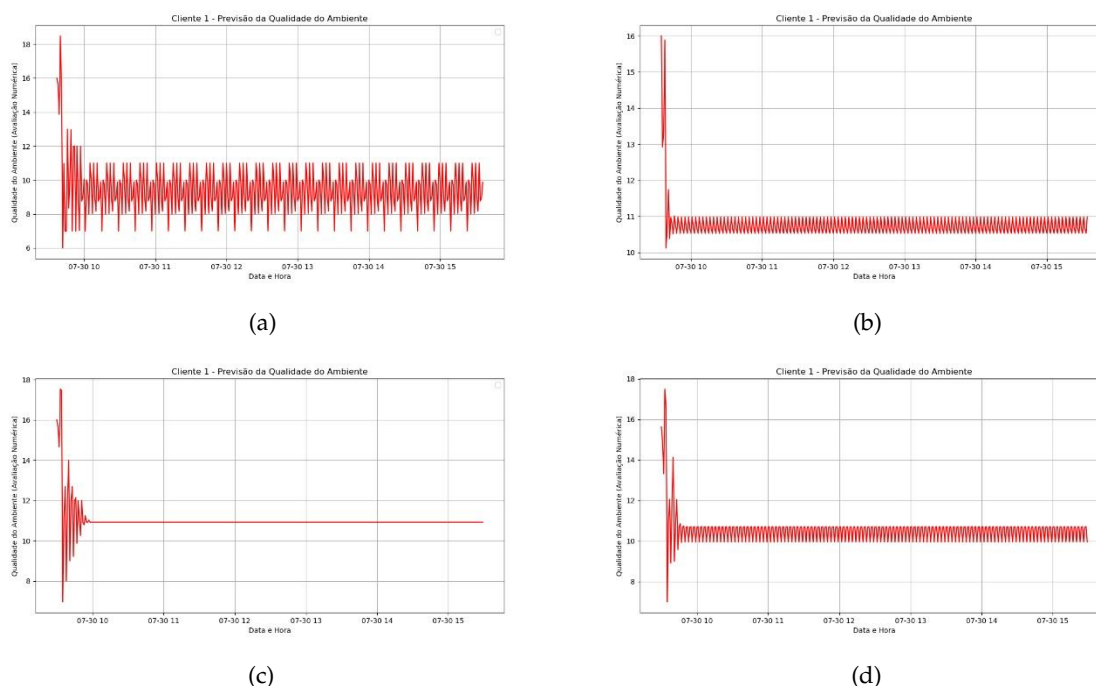


Figura 38 - Random Forest - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min

Ao analisar a figura 38 em comparação com os dados reais da figura 36, é possível concluir que nenhuma das previsões se ajusta às variações reais dos dados. A previsão para uma amostragem de 1 minuto é a que se aproxima mais dos dados reais, embora não seja perfeita. Todas as amostragens tendem a estabilizar em torno de uma avaliação da qualidade do ambiente de 11, que é aproximadamente a média dos valores reais. No entanto, a previsão obtida para uma amostragem de 10 minutos foi a menos precisa, dado que a partir das 11 horas a previsão estabiliza completamente, o que diverge dos dados reais. Além disso, todas as previsões apresentaram uma avaliação superior a 16, no início da previsão, o que não condiz com os dados reais. Em resumo, embora as previsões sejam geralmente boas, pois estão dentro da gama de valores esperados, exceto no início da previsão, o modelo falhou em capturar padrões

precisos nos dados. No entanto, não se trata necessariamente de um mau resultado. Segue-se a análise ao modelo ExtraTrees.

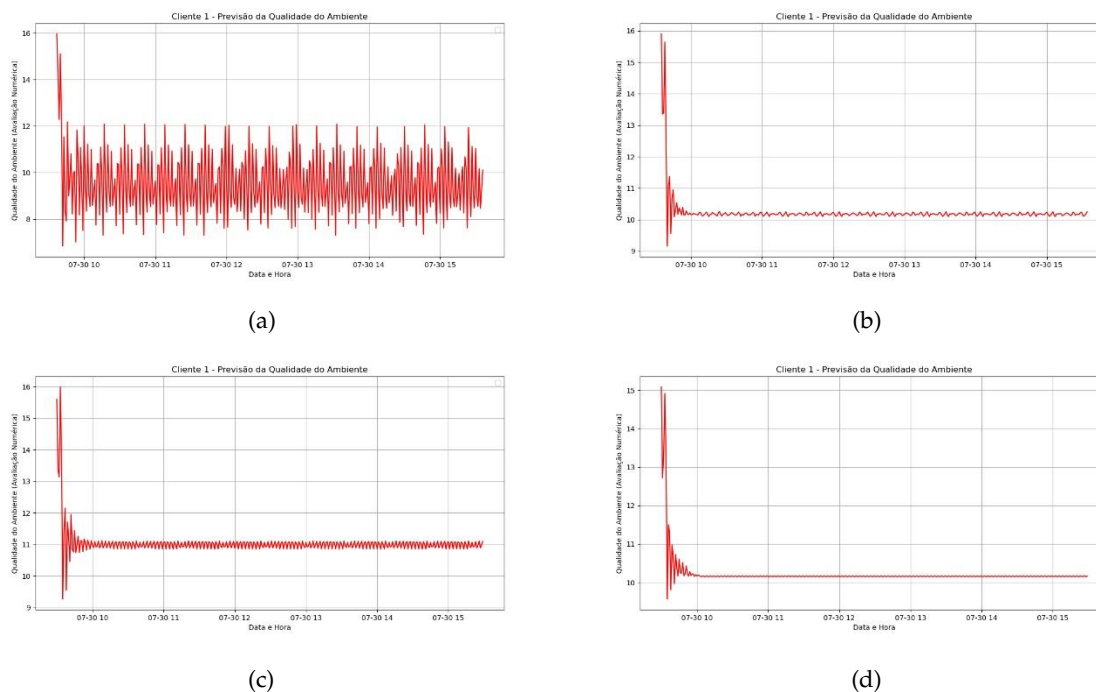
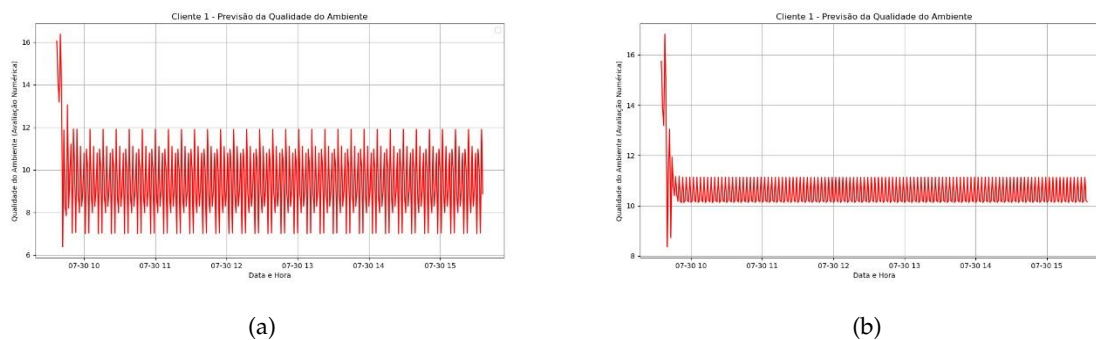
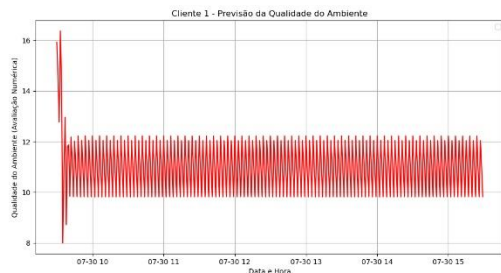


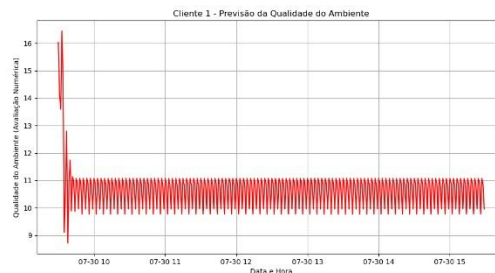
Figura 39 - ExtraTrees - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min

Em análise à figura 39, observa-se que, tal como o modelo Random Forest, apenas a previsão para uma amostragem de 1 minuto parece se aproximar mais dos dados reais. Nota-se também a existência de um pico na avaliação da qualidade do ambiente no início das previsões, o que não corresponde com os dados reais. Para as amostragens de 5, 10 e 15 minutos, a previsão parece também estabilizar entre 10 e 11, correspondente aproximadamente à média dos valores reais. Conclui-se que este modelo, demonstrou resultados piores que o Random Forest, apresentando resultados razoáveis, dado que não se adapta bem à variação dos dados exibida na figura 36 (a). Segue-se a análise ao modelo xGBoost.





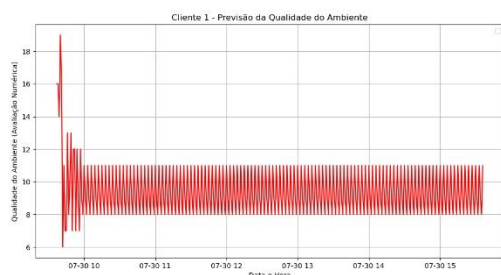
(c)



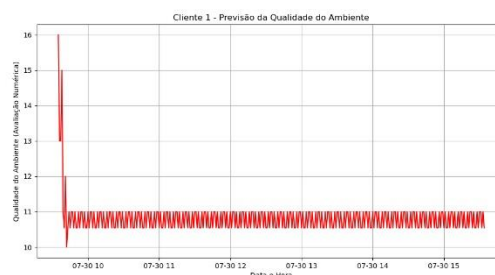
(d)

Figura 40 - xGBoost - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min

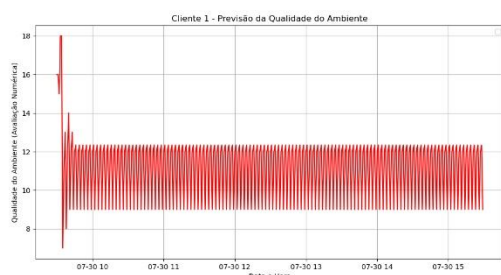
Em análise à figura 40, observa-se que, ao contrário dos dois modelos anteriores, o modelo xGBoost apresenta uma maior variação dos dados, para todas as amostragens, numa gama que se aproxima dos dados reais, embora com alguns pontos que não correspondem exatamente aos dados reais. No entanto, em comparação com os modelos anteriores, o xGBoost parece se adaptar melhor à variação dos dados, ainda que essa adaptação não seja perfeita. Segue-se a análise ao modelo Gradient Boosting.



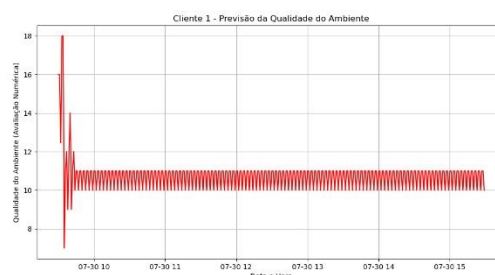
(a)



(b)



(c)



(d)

Figura 41 – Gradient Boosting - Previsão para cada amostragem: (a) 1 min; (b) 5 min; (c) 10 min; (d) 15 min

Em análise à figura 41, é possível observar que, tal como o modelo xGBoost, este modelo apresentou uma boa gama de valores, principalmente para as amostragens de 1 e 10 minutos. Tal como os modelos anteriores, este modelo também falhou na adaptação à variação dos dados, e apresenta na mesma um pico no início das previsões o que não vai de acordo com os dados reais.

Em suma, os modelos de regressão apresentaram excelentes resultados em termos de função de perda e apresentaram previsões bastante consistentes, com destaque para os modelos xGBoost e Gradient Boosting. No entanto, todas as previsões apresentaram uma variação nos dados alta, o que não vai de acordo com os dados reais. Ou seja, em todos os modelos observou-se uma oscilação entre 10 e 12, aproximadamente, a uma velocidade que não corresponde aos padrões reais, como é possível observar pela figura 36 (a). Concluindo, no caso dos modelos de regressão estes fornecem uma base sólida tanto na perda como na previsão, sendo que a última é longe de perfeita. Por outro lado, o modelo Facebook Prophet teve um desempenho inferior tanto em termos de função de perda como na previsão. Na próxima secção, serão apresentados os resultados dos modelos de *Deep Learning*, para comparar o desempenho com os dados reais, os modelos de regressão e com o Prophet.

5.4 Modelos de Deep Learning

Nesta subsecção, serão apresentados os resultados dos modelos de *Deep Learning*, nomeadamente LSTM, GRU e RNN. As simulações destes modelos não dependem apenas da amostragem, mas também do tamanho de uma sequência de valores. Os modelos de *Deep Learning* utilizam um parâmetro chamado ‘**input_shape**’, que define a estrutura dos dados de entrada que o modelo receberá. O ‘**input_shape**’ é crucial para determinar como os dados serão processados e envolve vários parâmetros que podem ser ajustados:

- **Batch Size:** Número de registos a processar antes de atualizar os parâmetros do modelo. Mantém-se constante durante as simulações.
- **Tamanho da Sequência de Input:** Refere-se ao número de passos de tempo que o modelo considerará para prever os valores futuros.
- **Número de Features:** Quantidade de variáveis independentes em cada intervalo de tempo. Este valor também é mantido constante nas simulações.

Em todas as simulações, o batch size e o número de features são constantes, o que é alterado é a amostragem dos dados e o tamanho da sequência de input. A otimização destes parâmetros é fundamental para melhorar o desempenho dos modelos, tanto em termos de duração quanto na minimização da função da perda. Para além destes parâmetros, são ainda apresentados os resultados obtidos em diferentes cenários:

- **Cenário 1:** Simulações com 3 clientes criados pelo PC.

- **Cenário 2:** Simulações com 3 Raspberry Pi's.

Em cada cenário, serão apresentados resultados com amostragens de 1, 5, 10 e 15 minutos, correspondendo a sequências de input de 60, 300, 600 e 900, respetivamente. Adicionalmente, para as amostragens de 10 e 15 minutos, serão testadas variações dos modelos de *Deep Learning* com o parâmetro '**return_sequences**' configurado como ('False') e ('True'), além das diferentes amostragens. O parâmetro '**return_sequences**' determina se o modelo deve devolver a sequência completa de saídas, ou seja, uma saída para cada intervalo de tempo da entrada ('True'), ou apenas a última saída ('False'). A sequência de saída, neste caso, refere-se ao conjunto de previsões gerados para cada instante da sequência de entrada. Quando configurado como ('False'), a execução do modelo é otimizada, resultando em tempos de simulação mais curtos devido à redução da quantidade de informações processadas. No entanto, o desempenho, em termos da função de perda, pode não ser tão bom quanto nas situações em que o parâmetro é definido como ('True'), pois não tem acesso a todas as sequências de saída, apenas a última. Nas tabelas seguintes, serão apresentados os resultados obtidos para cada modelo de *Deep Learning*, considerando os diferentes cenários, configurações de amostragem e tamanhos de sequência. Em todas as simulações, foram considerados:

- **Batch size** - 32
- **Nº de features** – 5 (Temperatura, Humidade, Qualidade do Ar, Ruído e Iluminância)
- **Métrica de avaliação da perda** – MSE
- **Tamanho da previsão futura** – 360 pontos (equivalente a 6 horas)

Nas subsecções seguintes serão apresentados os resultados obtidos de cada um dos cenários para as suas diferentes amostragens. Em primeiro lugar, serão apresentados os resultados práticos da função de perda agregada (servidor), que refletem a soma dos valores da perda nos diferentes clientes. Depois, será apresentado o gráfico da função de perda, que mostra a variação da perda ao longo do treino, analisada em cada *epoch*. Este gráfico permite visualizar como o modelo foi ajustando a sua performance durante o processo de treino. Por fim, serão exibidos os resultados gráficos da previsão, considerando um período de 6 horas. Este gráfico ilustra o desempenho preditivo do modelo em diferentes cenários de amostragem. Para evitar que o documento fique sobrecarregado de imagens, os resultados apresentados nas subsecções seguintes serão referentes apenas ao cliente 1.

5.4.1 Cenário 1 – Amostragem de 1 minuto

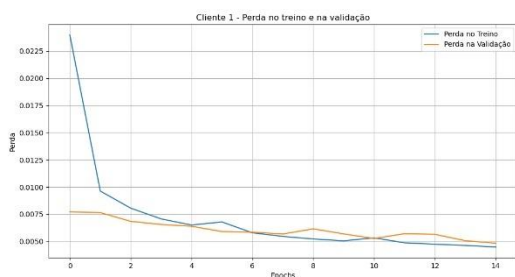
Nesta sub-subsecção serão apresentados os resultados obtidos dos modelos de *Deep Learning*, tendo em conta uma amostragem de 1 minuto. A sequência de input utilizada foi de

1440 (referente apenas a 1 dia, dadas as limitações de memória). Na tabela 14, estão apresentados os resultados obtidos.

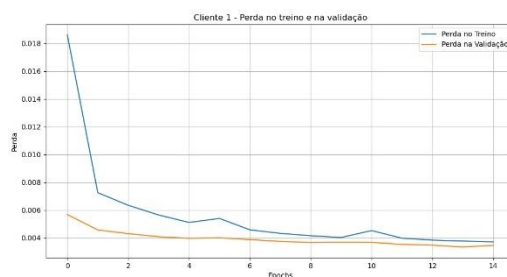
Tabela 14 - Resultados Cenário 1: Amostragem 1 minuto

Modelo	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
LSTM	17:47:13	0.016488	0.014817	0.016241
GRU	19:11:54	0.015040	0.010864	0.011739
RNN	9:52:27	0.023468	0.019075	0.016721

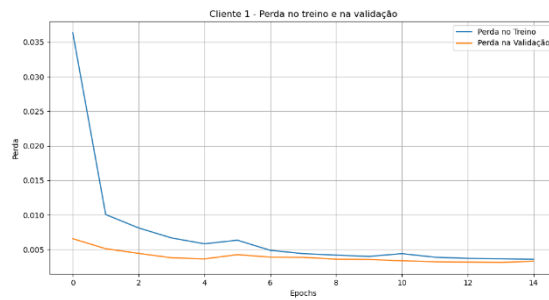
Como é possível observar pela tabela 14, a duração da execução de cada modelo é bastante alta. Além disso, os resultados em termos de perda foram bastante superiores aos valores de perda dos modelos de regressão, indicando que para esta amostragem os modelos obtidos foram de longe os melhores. Na figura 42, é apresentada a variação da perda a cada *epoch*, tanto no treino como na validação. A perda no treino reflete o ajuste do modelo aos dados de treino, enquanto a perda na validação avalia o desempenho do modelo em dados que ainda foram dados ao modelo. Idealmente, estas perdas devem estar próximas, pois isso indica que o modelo está a generalizar bem para dados desconhecidos. Ou seja, o modelo está a alcançar um bom equilíbrio entre o desempenho nos dados de treino e a capacidade de generalização para novos dados.



(a)



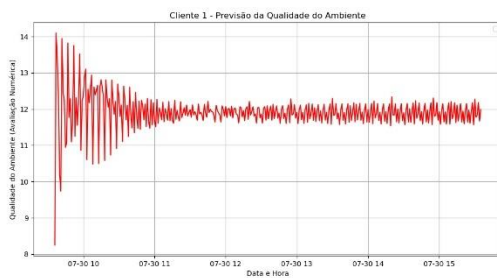
(b)



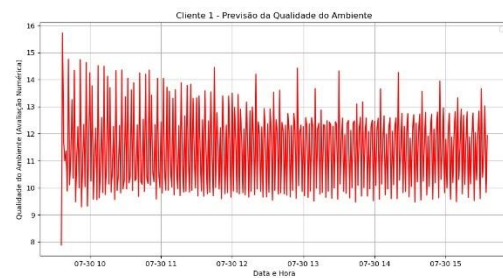
(c)

Figura 42 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

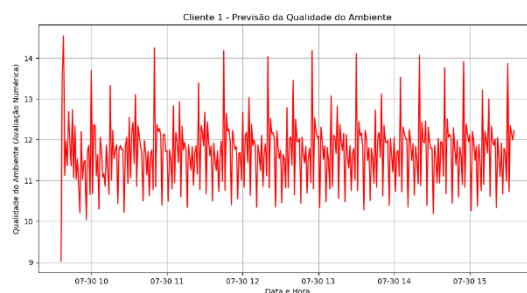
Em análise à figura 42, que a função da perda obtida para o cliente 1 foi bastante baixa, tanto na validação como no treino, para todos os modelos de DL. Contudo, a perda agregada pelo servidor, exibida na tabela 14, foi superior à perda apresentada pelo cliente, indicando que houve pelo menos 1 cliente que não se ajustou bem aos dados. Para este caso, o pior dos clientes foi o cliente 3. Na figura 43, estão ilustrados os resultados obtidos quanto à previsão de um período de 6 horas.



(a)



(b)



(c)

Figura 43 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Em análise à figura 43, é possível concluir que, os modelos de *Deep Learning* apresentaram um desempenho satisfatório. Nenhuma das previsões apresentada, parece ter seguido idealmente o comportamento dos dados reais, contudo o GRU e o RNN foram os que mais se aproximaram. Os resultados ficaram aquém do esperado, dado que se espera que com uma amostragem menor o modelo se adaptasse melhor aos dados, mas isso não aconteceu. Outro

fator negativo em relação a este caso, é a sua duração, não sendo esta viável para aplicações do mundo real.

Dado que se trata do cenário 1, onde todos os clientes são geridos pelo servidor, a duração no cenário 2 (com os Raspberry Pi como clientes) seria ainda maior, o que não é ideal. Para o caso do cenário 2, o tempo de processamento era excessivamente maior do que o tempo apresentado no cenário 1. Tendo em conta a elevada duração deste cenário, optou-se por aumentar a amostragem de 1 para 5 minutos, de forma a reduzir tanto a duração da execução do modelo, bem como tentar melhor a previsão apresentada pelos modelos GRU e RNN, na figura 43. Esta alteração pode levar a aumentos na função de perda devido à menor granularidade dos dados, mas é necessária para tornar as simulações mais viáveis em termos de duração. De seguida são apresentados os resultados obtidos no cenário 1 para uma amostragem de 5 minutos.

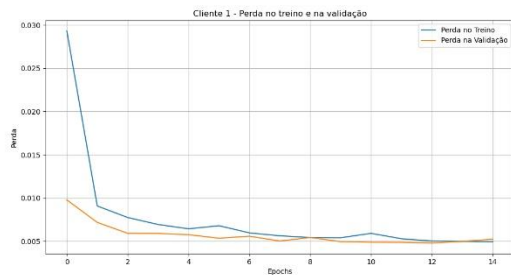
5.4.2 Cenário 1 – Amostragem de 5 minutos

Nesta subsecção serão apresentados os resultados obtidos dos modelos de *Deep Learning*, tendo em conta uma amostragem de 5 minutos. A sequência de input utilizada foi de 1440 (referente a 5 dias). Na tabela 15, estão apresentados os resultados obtidos.

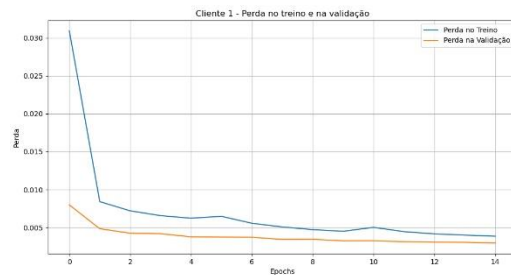
Tabela 15 - Resultados Cenário 1: Amostragem 5 minutos

Modelo	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
LSTM	3:32:47	0.010285	0.011682	0.010204
GRU	4:02:25	0.014328	0.009919	0.009073
RNN	1:59:07	0.012785	0.0075317	0.009787

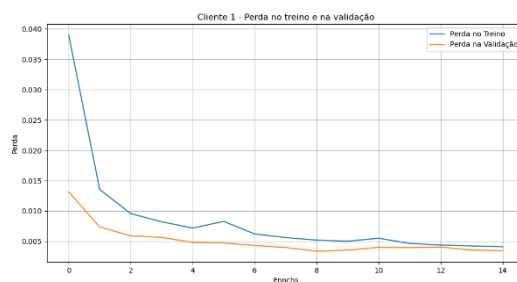
Ao comparar as tabelas 14 e 15, observa-se uma redução significativa no tempo de treino dos modelos, com uma diminuição de até 5 vezes no tempo de processamento. Além disso, esse ganho em eficiência temporal foi acompanhado por uma diminuição na função de perda: 37,5% no modelo LSTM e cerca de 19% no modelo GRU e 40% no RNN. Na figura 44, estão ilustrados os resultados gráficos da perda referente a cada modelo.



(a)



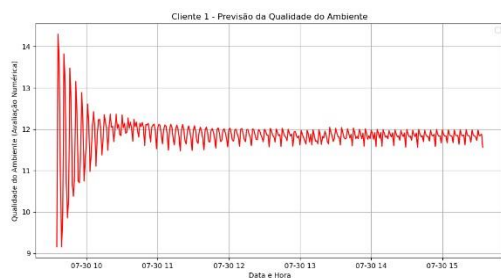
(b)



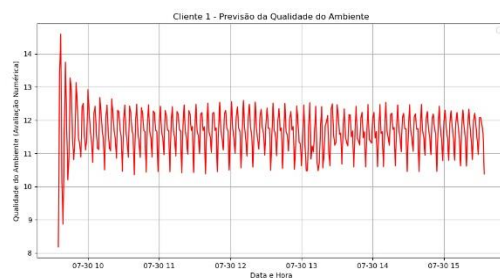
(c)

Figura 44 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

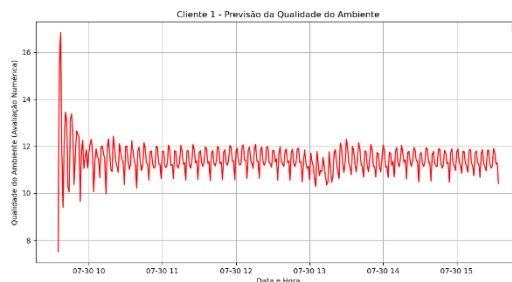
Em análise à figura 44, é possível concluir a perda se manteve praticamente constante em todos os modelos comparando com a figura 43. Em análise às tabelas 14 e 15, é possível concluir para estes conjuntos de dados, possivelmente a amostragem de 1 minuto capturou mais variações instantâneas ou ruído nos dados, o que pode ter dificultado a modelação, resultando numa perda agregada maior. Uma amostragem maior (5 minutos) pode facilitar o ajuste a padrões temporais mais consistentes, o que resultou numa perda agregada menor. Contudo, o cliente 1 apresentou resultados semelhantes em termos de perda para as duas amostragens. Na figura 45, são apresentados os resultados em relação à previsão.



(a)



(b)



(c)

Figura 45 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Ao comparar as figuras 43 e 45, referentes à previsão para uma amostragem de 1 e 5 minutos respetivamente, é possível observar uma diminuição da variância em todos os modelos, o que não corresponde ao comportamento exibido pelos dados reais, figura 36(a). Comparativamente aos dados reais, o modelo GRU é o que apresenta os melhores resultados, tanto pela gama de valores que apresenta, bem como a sua variação temporal que se assemelha à dos dados reais. Todos os modelos para esta amostragem, exibem também um pico no início da previsão, o que também não corresponde com os dados reais, contudo todas elas começam num ponto bastante próximo dos valores reais. O modelo LSTM parece ser o único modelo capaz de detetar alguns padrões no início da previsão, dado que não apresenta uma subida tão acentuada no início da previsão. Antes de estabilizar e não considerando o pico existente, a subida da sua previsão não é tão acentuada perto das 10h, como ocorre nos outros modelos. No entanto, após alcançar o valor médio da previsão, o modelo apresenta resultados menos favoráveis. A seguir, serão apresentados os resultados obtidos no cenário 2 com uma amostragem de 5 minutos.

5.4.3 Cenário 2 – Amostragem de 5 minutos

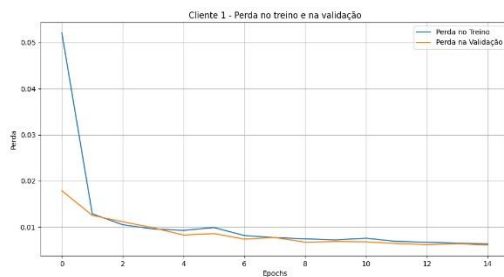
Nesta subsecção serão apresentados os resultados obtidos dos modelos de *Deep Learning*, tendo em conta uma amostragem de 5 minutos. A sequência de input utilizada foi de 1440 (referente a 5 dias). Na tabela 16, estão apresentados os resultados obtidos.

Tabela 16 - Resultados Cenário 2: Amostragem 5 minutos

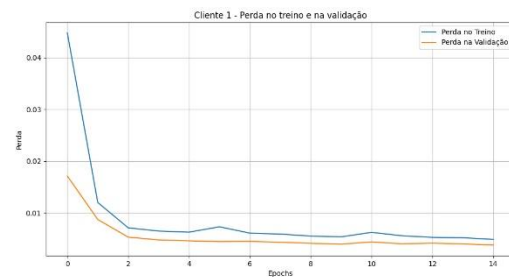
Modelo	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
LSTM	7:26:51	0.014328	0.009919	0.009146
GRU	8:55:45	0.009782	0.008870	0.008268

RNN	4:39:55	0.016740	0.009321	0.008693
-----	---------	----------	----------	----------

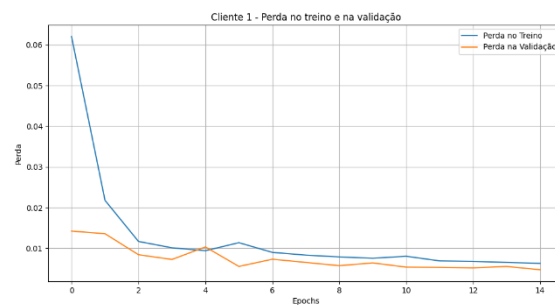
Ao comparar os valores das tabelas 15 e 16, verifica-se que a função de perda teve uma pequena melhoria, tendo diminuído uma milésima praticamente em todos os modelos, mas a duração da execução aumentou em 2,5 vezes devido ao menor poder de computação do Raspberry Pi. É de esperar que a função de perda permaneça relativamente estável, uma vez que não é diretamente influenciada pelo dispositivo de execução. Na figura 46, estão ilustrados os resultados obtidos em termos da perda para cada modelo.



(a)



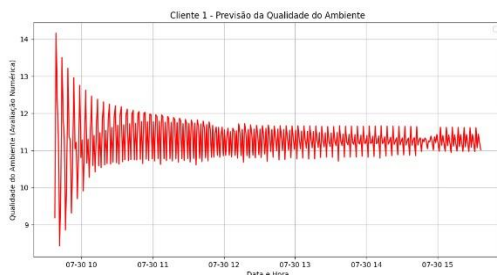
(b)



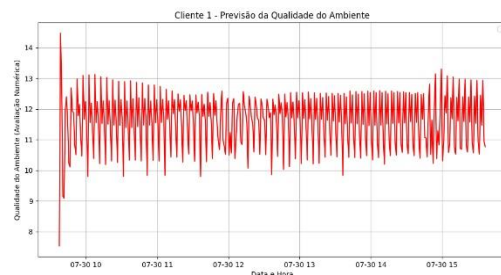
(c)

Figura 46 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

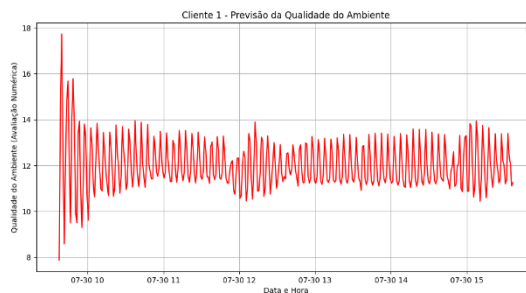
Ao comparar as figuras 44 e 46, é possível concluir que as perdas apresentadas de todos os modelos foram bastante semelhantes. A perda agrupada pelo servidor não sofreu grande alteração, conforme evidenciado pela tabela 16. Seguem-se os resultados gráficos referentes à previsão, figura 47.



(a)



(b)



(c)

Figura 47 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Ao comparar as figuras 45 e 47, referentes aos cenários 1 e 2 da previsão para uma amostragem de 5 minutos, respectivamente, é possível observar um ligeiro aumento da variância em todos os modelos, o que está mais de acordo com os dados reais. Comparativamente aos dados reais, as previsões dos modelos da figura 47, já se assemelham mais ao comportamento dado pelos dados reais. No entanto, o modelo LSTM, ainda não produziu os melhores resultados, pois a sua gama de valores torna-se bastante constante a partir de um certo ponto, o que não reflete adequadamente com os dados reais. O modelo RNN melhorou também na detecção de alguns padrões, visto que já não tem uma subida tão acentuada perto das 10h, o que vai de acordo com os dados reais. Dado o elevado tempo de treino do modelo, será aumentada a amostragem de 5 para 10 minutos para avaliar melhor o impacto na duração, na função de perda e na previsão.

5.4.4 Cenário 1 – Amostragem de 10 minutos

Nesta subsecção serão apresentados os resultados obtidos dos modelos de *Deep Learning*, tendo em conta uma amostragem de 10 minutos. A sequência de input utilizada foi de 720 (referente a 5 dias) pois, com uma amostragem de 10 minutos tem-se 6 valores por hora, 144 diariamente e 720 a cada 5 dias. Na tabela 17, estão apresentados os resultados obtidos.

Tabela 17 - Resultados Cenário 1: Amostragem 10 minutos

Modelo	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
LSTM	00:49:05	0.017261	0.010858	0.008998
GRU	00:58:26	0.005507	0.006297	0.005174
RNN	00:28:07	0.013279	0.008688	0.007034

Como se pode observar na tabela 17, houve uma redução significativa no tempo de treino dos modelos. Além disso, os modelos GRU e RNN demonstraram uma melhoria significativa na função de perda em comparação com os resultados da tabela 15, principalmente nos modelos GRU e RNN. Na figura 48, estão apresentados os resultados da perda para cada um dos modelos.

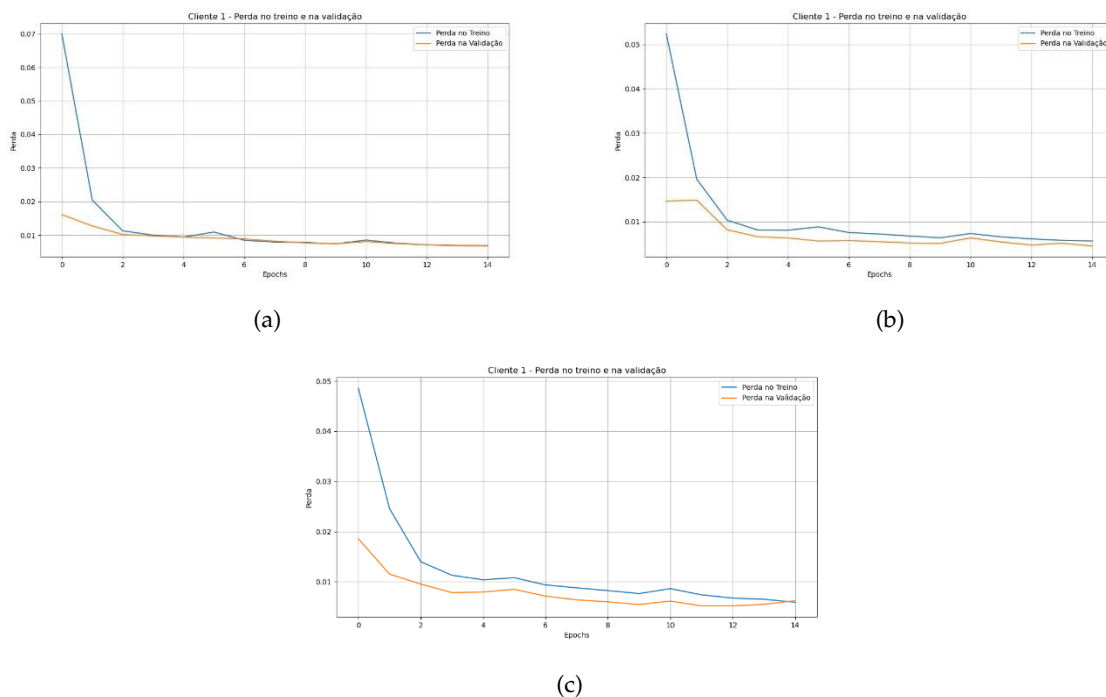
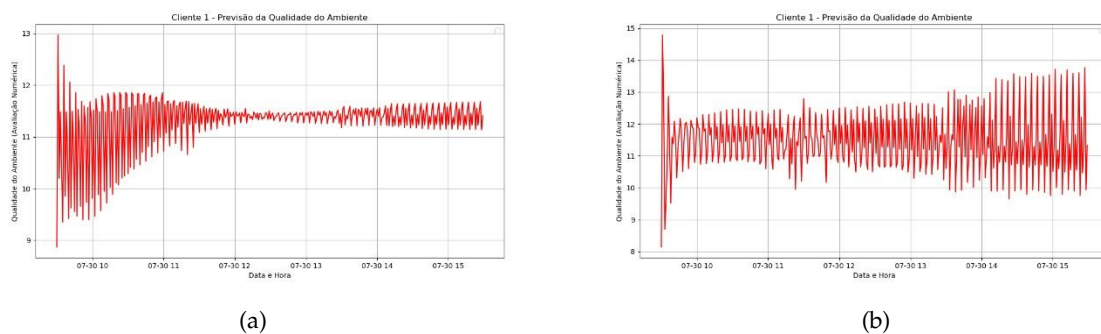
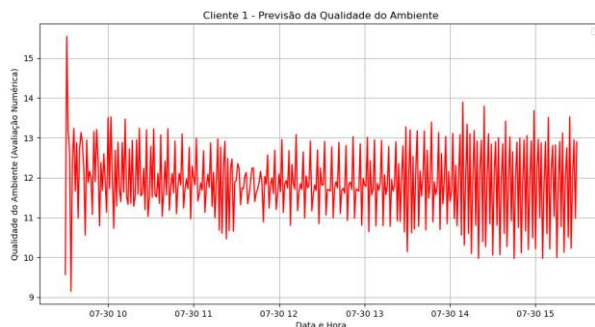


Figura 48 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Comparando as figuras 44 e 48, referentes ao cenário 1 com uma amostragem de 5 e 10 minutos, respetivamente, é possível observar no cliente 1 a perda não sofreu grande alteração, tendo-se apenas notado num pequeno aumento no modelo RNN. Notar que os gráficos da perda são referentes à perda de um cliente em específico, enquanto a perda apresentada nas tabelas se refere à perda agrupada após cada iteração de comunicação entre clientes e servidor. Seguem-se os gráficos das previsões, figura 49.





(c)

Figura 49 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Comparando as figuras 45 e 49, referentes ao cenário 1 com uma amostragem de 5 e 10 minutos, respectivamente, é possível observar um aumento geral na variância, adequada ao comportamento dos dados reais. Em termos de gamas das previsões, os modelos GRU e RNN foram os que apresentaram melhores resultados. Contudo, no início das previsões, o modelo LSTM apresenta a previsão mais próxima dos dados reais, dado que a subida inicial não é tão acentuada. Para este cenário o aumento da amostragem não só diminuiu o tempo de computação, como também melhorou a precisão das previsões dos modelos. Serão agora apresentados os resultados obtidos para o cenário 2, referente a uma amostragem de 10 minutos.

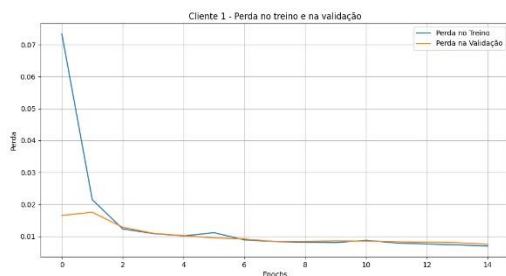
5.4.5 Cenário 2 – Amostragem de 10 minutos

Nesta subsecção serão apresentados os resultados obtidos dos modelos de *Deep Learning*, tendo em conta uma amostragem de 10 minutos. A sequência de input utilizada foi de 720 (referente a 5 dias). Na tabela 18, estão apresentados os resultados obtidos.

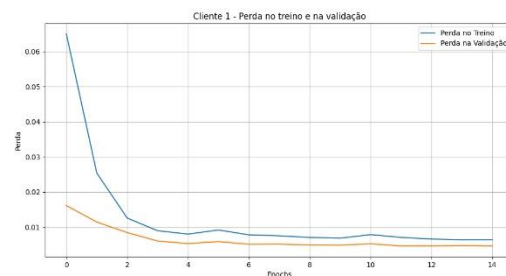
Tabela 18 - Resultados Cenário 2: Amostragem 10 minutos

Modelo	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
LSTM	2:16:01	0.010914	0.010266	0.009609
GRU	2:42:42	0.006645	0.005632	0.005373
RNN	1:09:17	0.01428	0.007042	0.006081

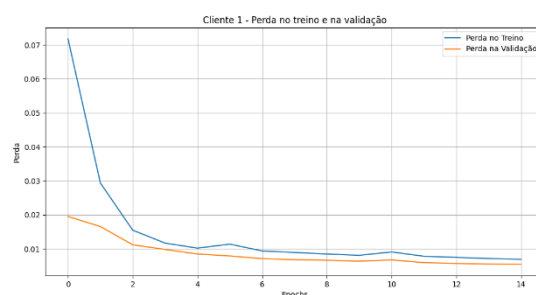
Com base nos resultados observados, é possível concluir que a duração do treino dos modelos aumentou em comparação com a tabela 17, referente ao cenário 1 com uma amostragem de 10 minutos. No entanto, houve uma diminuição significativa na função de perda no modelo RNN. Na figura 50, estão apresentados os resultados gráficos da perda.



(a)



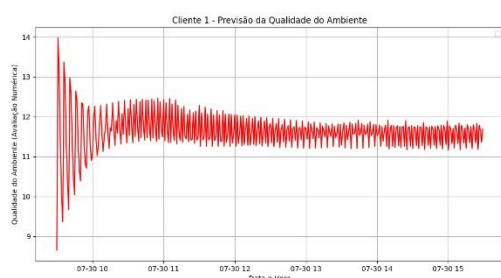
(b)



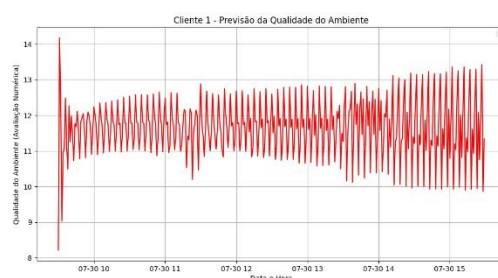
(c)

Figura 50 - Perda no treino e na validação dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

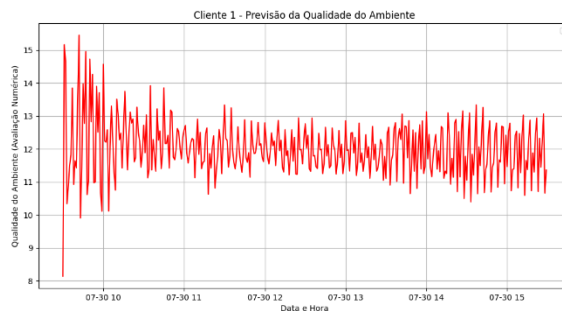
Comparando as figuras 46, 48 e 50, que mostram a perda no cenário 2 com uma amostragem de 5 minutos e nos cenários 1 e 2 com uma amostragem de 10 minutos, é possível observar que a perda nas amostragens de 10 minutos se manteve relativamente constante. Sendo que para o cliente 1, a perda parece ter sido ligeiramente inferior para uma amostragem de 5 minutos, segundo a figura 46. Quanto à perda agregada dos clientes, a amostragem de 10 minutos obteve os melhores resultados, sendo que o cenário 1 obteve melhores resultados nos modelos LSTM e GRU, enquanto o cenário 2 apresentou uma perda mais baixa no modelo RNN. Na figura 51, estão representadas as previsões referentes a cada modelo.



(a)



(b)



(c)

Figura 51 - Previsão para cada um dos modelos de Deep Learning: (a) LSTM; (b) GRU; (c) RNN

Comparando as figuras 47, 49 e 51, referentes ao cenário 2 com uma amostragem de 5 minutos e aos cenários 1 e 2 com uma amostragem de 10 minutos, respetivamente, é possível observar que no cenário 2 para uma amostragem de 10 minutos a previsão do modelo RNN piorou, apresentando discrepâncias com os dados reais no início da previsão. A previsão do LSTM melhorou em relação ao cenário 2 para uma amostragem de 5 minutos, dado a sua subida na parte inicial da previsão, embora continue a apresentar os mesmos problemas após as 11h, quando a previsão estabiliza. Para este último cenário, o modelo GRU apresentou resultados bastante interessantes, semelhantes aos resultados obtidos para o cenário 1.

Importa salientar que os modelos foram treinados em dispositivos Raspberry Pi. A performance da função de perda não é diretamente influenciada pelo hardware, mas sim pelos parâmetros de modelagem e pela granularidade da amostragem. Assim, uma amostragem de 10 minutos pode ser uma opção viável, mas ainda existe potencial para otimizar a duração do treino dos modelos sem comprometer demasiado a função de perda. Na próxima subsecção, serão testadas amostragens de 15 minutos e diferentes arquiteturas de modelos de *Deep Learning*.

5.4.6 Outras Arquiteturas (Amostragens de 10 e 15 minutos)

Em cada modelo, serão avaliadas 5 situações distintas, conforme descrito na tabela 19. Esta tabela detalha a amostragem, o tamanho da sequência de entrada, o tipo de modelo a utilizar (RNN, GRU ou LSTM), o número de camadas do modelo (incluindo camadas densas), o número de neurónios em cada camada e a configuração da opção `return_sequences`. Para as novas arquiteturas, foi usada apenas uma camada do modelo, de forma a reduzir a complexidade do processamento sequencial, e foi ainda aumentado o número de camadas Dense, em

relação à “Referência”, de forma a aumentar ainda mais a capacidade do modelo de captar padrões não lineares após o processamento de cada sequência. Importa destacar que a situação “Referência” corresponde à configuração utilizada nas simulações anteriores. O objetivo destas simulações é analisar o impacto desses parâmetros no treino dos modelos, tanto em termos da função de perda quanto na duração do treino.

Tabela 19 - Situações a simular com as diferentes arquiteturas para cada modelo de Deep Learning

	Amostragem (minutos)	Tamanho Sequência	Modelo		Dense		Return Sequences
			N. ° Camadas	Neurónios	N. ° Camadas	Neurónios	
Referência	10	720 (5 dias)	2	64, 32	1	1	True, False
Situação 1	10	720 (5 dias)	1	64	2	32, 1	False
Situação 2	10	720 (5 dias)	1	64	3	64, 32, 1	False
Situação 3	15	480 (5 dias)	2	64, 32	1	1	True, False
Situação 4	15	480 (5 dias)	1	64	2	32, 1	False
Situação 5	15	480 (5 dias)	1	64	3	64, 32, 1	False

Serão agora apresentados todos os resultados obtidos para cada modelo em diferentes cenários e situações. Cada sub-subsecção será referente a um modelo, LSTM, RNN ou GRU, onde os resultados referentes a cada cenário serão apresentados. Quanto aos gráficos da perda e das previsões, serão apresentados os melhores resultados de cada cenário, bem como os resultados da “Situação 3”, referente a uma amostragem de 15 minutos, onde o modelo usa a mesma arquitetura utilizada na “Referência. Esta arquitetura foi usada para simular os casos apresentados anteriormente.

5.4.6.1 RNN – Resultados

Nesta sub-subsecção serão apresentados os resultados obtidos do modelo RNN, tendo em conta as situações apresentadas na tabela 19. As tabelas 20 e 21 mostram os resultados para o modelo RNN nos cenários 1 e 2, respetivamente.

Tabela 20 - Resultados RNN: Cenário 1

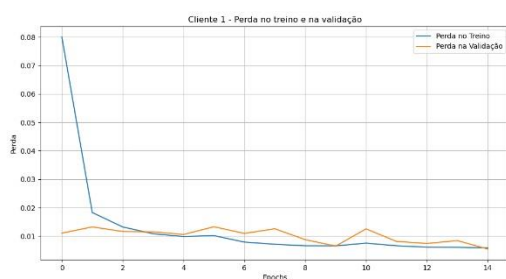
	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
Referência	0:28:07	0.013278738	0.008688404	0.007034
Situação 1	0:12:31	0.011689955	0.007662599	0.005479
Situação 2	0:12:27	0.010875766	0.005981457	0.005882
Situação 3	0:13:13	0.019831129	0.011509807	0.008143
Situação 4	0:05:12	0.013149577	0.007407959	0.00646

Situação 5	0:05:05	0.012690663	0.009373925	0.008381
------------	---------	-------------	-------------	----------

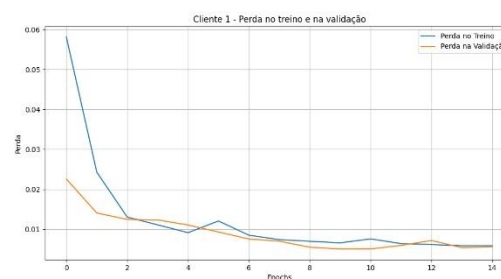
Tabela 21 - Resultados RNN: Cenário 2

	Duração (Horas:Minutos)	MSE		
		Iteração 1	Iteração 2	Iteração 3
Referência	1:09:17	0.014280049	0.007041827	0.006081
Situação 1	0:25:19	0.018170633	0.010645097	0.006197
Situação 2	0:26:05	0.012009371	0.006389154	0.006187
Situação 3	0:28:56	0.012573609	0.007643979	0.006442
Situação 4	0:11:57	0.018809385	0.008380766	0.006899
Situação 5	0:12:07	0.011913318	0.007773381	0.007455

Em análise às tabelas 20 e 21, é possível observar que a duração do treino no cenário 2 é mais do dobro do tempo do cenário 1, devido às limitações computacionais dos Raspberry Pi's. Relativamente à perda, comparando o cenário 1 com o cenário 2, ambas apresentaram 3 situações em que a perda foi menor, sendo que a mais baixa foi obtida para a "Situação 1" relativa ao cenário 1. Com base nos resultados das simulações, é possível observar que a configuração **'return_sequences = False'** reduz a duração do treino dos modelos, e nem sempre influencia negativamente a função da perda, à exceção da "Situação 5" referente ao cenário 1 e nas situações 1, 2, 4 e 5 do cenário 2. Contudo o aumento nunca é significativo. Dada a redução do tempo de treino, e visto que a perda nunca aumenta significativamente, a configuração **'return_sequences = False'** torna-se uma opção viável para as previsões. A figura 52 apresenta os melhores resultados da perda para cada cenário. No cenário 2, o melhor resultado corresponde à "Referência" já discutido na subsecção 5.4.4. Assim a figura 52, irá apresentar os resultados da "Situação 1" uma vez que foi o melhor resultado no cenário 1, e no cenário 2, obteve um desempenho bastante próximo. Na figura 53, serão apresentados os resultados relativos a uma amostragem de 15 minutos, "Situação 3".

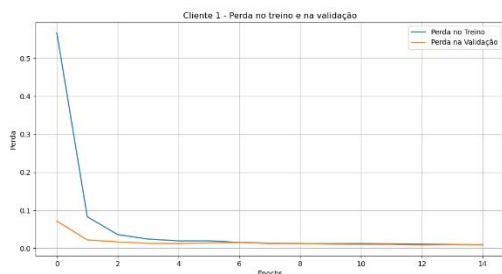


(a)

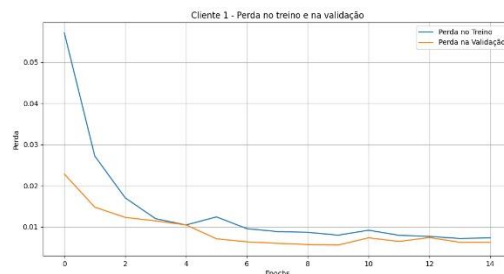


(b)

Figura 52 - Perda no treino e na validação do modelo RNN na Situação 1 para cada cenário: (a) Cenário 1; (b) Cenário 2



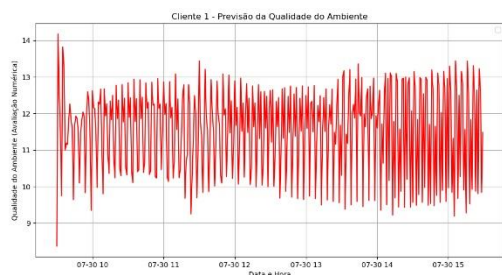
(a)



(b)

Figura 53 - Perda no treino e na validação do modelo RNN na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2

Em análise às figuras 52 e 53, é possível concluir que, a perda tanto no treino como na validação foram bastante baixas. Contudo, os valores da perda, com amostragens mais pequenas (5 e 10 minutos) apresentaram melhores resultados. Faltam apenas comparar as previsões. Seguem-se as previsões para cada cenário, para as situações 1 e 3 representadas nas figuras 54 e 55, respetivamente.



(a)

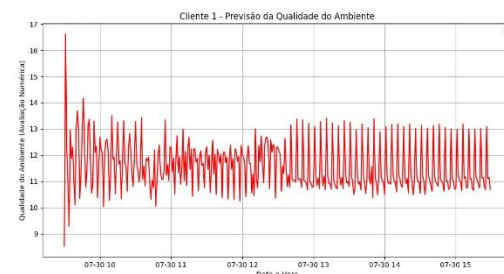


(b)

Figura 54 - Previsão dada pelo modelo RNN na Situação 1 em cada cenário: (a) Cenário 1; (b) Cenário 2



(a)



(b)

Figura 55 - Previsão dada pelo modelo RNN na Situação 3 em cada cenário: (a) Cenário 1; (b) Cenário 2

Ao analisar as figuras 54 e 55, é possível observar que a “Situação 1” apresentou os melhores resultados, tanto em termos de perda como na previsão, principalmente no cenário 1.

Já na “Situação 3”, os resultados obtidos, especialmente no cenário 1, ficaram aquém do esperado, devido à grande variação no início da previsão. Em ambas as situações para o cenário 2, os resultados obtidos foram satisfatórios, com exceção dos picos observados nas previsões. Segue-se a análise ao modelo LSTM.

5.4.6.2 LSTM - Resultados

Nesta sub-subsecção serão apresentados os resultados obtidos do modelo LSTM, tendo em conta as situações apresentadas na tabela 19. As tabelas 22 e 23, apresentam os resultados para o modelo LSTM nos cenários 1 e 2, respetivamente.

Tabela 22 - Resultados LSTM: Cenário 1

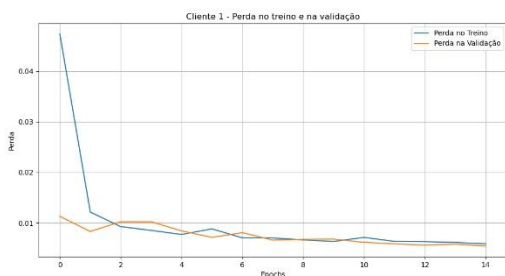
	Duração (hh:mm:ss)	MSE		
		Iteração 1	Iteração 2	Iteração 3
Standard	0:49:05	0.017261096	0.010857644	0.008998
Situação 1	0:25:06	0.014771755	0.006964202	0.006221
Situação 2	0:26:09	0.032694371	0.00808384	0.006661
Situação 3	0:23:05	0.012231191	0.010035698	0.009378
Situação 4	0:10:55	0.015344137	0.010774749	0.008651
Situação 5	0:11:59	0.062905159	0.006801977	0.007169

Tabela 23 - Resultados LSTM: Cenário 2

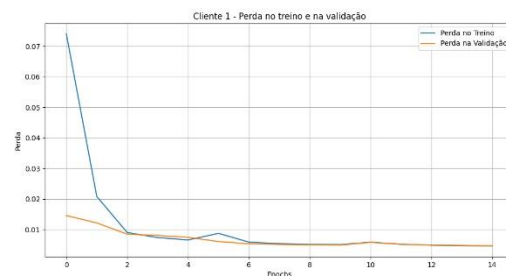
	Duração (hh:mm:ss)	Loss		
		Iteração 1	Iteração 2	Iteração 3
Standard	2:16:01	0.010914364	0.01026579	0.009609
Situação 1	1:08:54	0.020169728	0.008009917	0.007038
Situação 2	1:07:26	0.00839545	0.006653567	0.006866
Situação 3	1:06:18	0.014069369	0.012633858	0.010281
Situação 4	0:30:16	0.010384923	0.011022524	0.009562
Situação 5	0:30:32	0.008600255	0.008779376	0.007969

Em análise às tabelas 22 e 23, é possível observar que a duração do treino no cenário 2 é mais do dobro do tempo do cenário 1, devido às limitações computacionais dos Raspberry Pi's, evidenciado igualmente no modelo RNN. Relativamente à perda, o cenário 1 apresentou, em todos os casos, melhores resultados. Com base nos resultados das simulações, é possível observar que a configuração **'return_sequences = False'** reduz a duração do treino dos modelos, bem como a perda em todas as situações. Tal como no modelo RNN, serão apresentados

os gráficos da função da perda para os melhores casos, bem como para a “Situação 3”, referente a uma amostragem de 15 minutos. A figura 56, representa a perda tanto no treino como na validação, referente à “Situação 1” no cenário 1 e à “Situação 2” no cenário 2. Na figura 57, serão apresentados os resultados referentes à “Situação 3”.

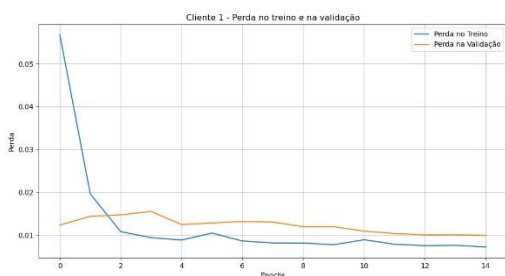


(a)

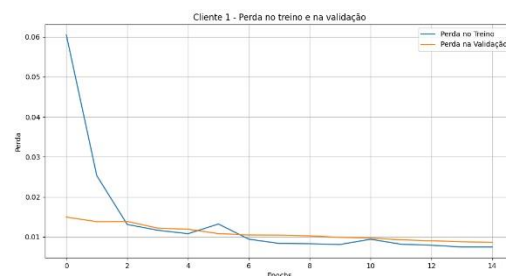


(b)

Figura 56 – Perda no treino e na validação do modelo LSTM: (a) Cenário 1 – Situação 1; (b) Cenário 2 – Situação 2



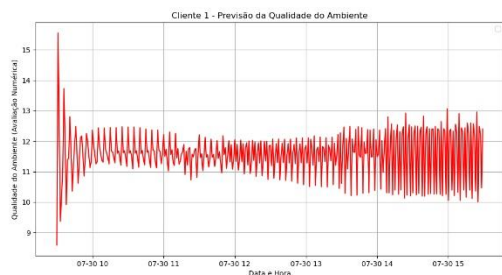
(a)



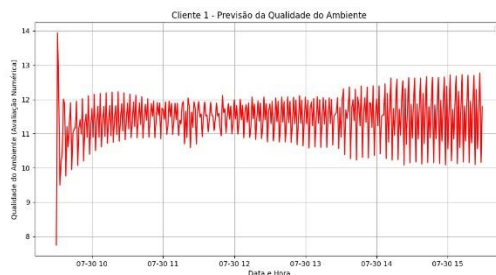
(b)

Figura 57 - Perda no treino e na validação do modelo LSTM na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2

Como é possível observar pelas figuras 56 e 57, a perda obtida na “Situação 2” foi a mais entre todas as amostragens, indicando que o parâmetro `‘return_sequences = False’` foi uma escolha acertada para este modelo. Por outro lado, na “Situação 3”, a perda obtida foi uma das mais elevadas entre as diferentes amostragens, sendo esta bastante semelhante à perda para uma amostragem de 5 minutos. Para compreender como a perda afeta o desempenho destes modelos, seguem-se as previsões para cada situação e cenário, representados nas figuras 58 e 59.



(a)



(b)

Figura 58 - Previsão dada pelo modelo LSTM: (a) Cenário 1 – Situação 1; (b) Cenário 2 – Situação 2

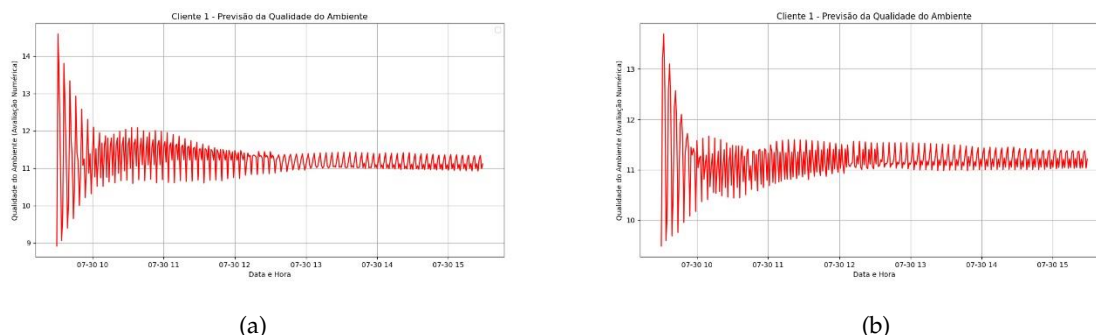


Figura 59 - Previsão dada pelo modelo LSTM na Situação 3 em cada cenário: (a) Cenário 1; (b) Cenário 2

Ao analisar as figuras 58 e 59, é evidente que os resultados da previsão obtidos nas Situação 1 e 2 foram superiores aos da “Situação 3”, o que já seria de esperar pela diferença nas perdas das situações em causa. Tanto a previsão da “Situação 1” como da “Situação 2”, foi bastante satisfatória, à exceção dos picos no início da previsão. Em ambos os cenários, os valores previstos estão dentro das gamas esperadas. Quanto à previsão da “Situação 3”, os resultados obtidos ficaram aquém do esperado, dado que não corresponde exatamente aos dados reais. Em comparação ao RNN, o LSTM segue mais a tendência inicial dos dados, dado que a subida da avaliação da qualidade do ambiente nas primeiras horas se alinha mais com os dados reais. No entanto, das 12h às 15h, o RNN oferece uma gama de valores mais precisa aos dados reais. No fim da previsão ambos são relativamente semelhantes. Segue-se a análise ao modelo GRU.

5.4.6.3 GRU - Resultados

Nesta sub-subsecção serão apresentados os resultados obtidos do modelo GRU, tendo em conta as situações apresentadas na tabela 19. As tabelas 24 e 25, referem-se ao modelo GRU, também nos cenários 1 e 2, respetivamente.

Tabela 24 - Resultados GRU: Cenário 1

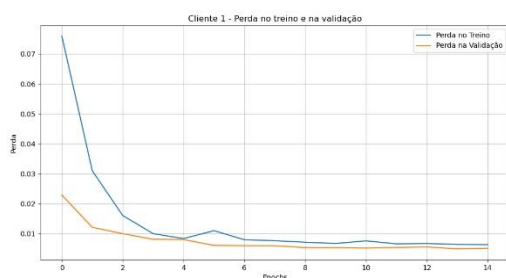
	Duração (Horas:Minutos)	Loss		
		Iteração 1	Iteração 2	Iteração 3
Standard	0:58:26	0.005507466	0.006296912	0.005174
Situação 1	0:36:42	0.012474868	0.007046854	0.007495
Situação 2	0:31:24	0.007383432	0.005174778	0.006491
Situação 3	0:26:13	0.00915303	0.00781554	0.006955

Situação 4	0:15:17	0.010369383	0.007201054	0.005918
Situação 5	0:14:17	0.00701909	0.005551226	0.006059

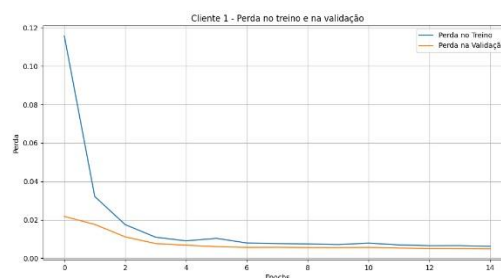
Tabela 25 - Resultados GRU: Cenário 2

	Duração (Horas:Minutos)	Loss		
		Iteração 1	Iteração 2	Iteração 3
Standard	2:42:42	0.006644989	0.005631638	0.005373
Situação 1	1:22:26	0.008187298	0.005319096	0.004629
Situação 2	1:20:27	0.00959261	0.006129577	0.004662
Situação 3	1:15:32	0.009020406	0.009471907	0.007296
Situação 4	0:38:05	0.009952938	0.007025705	0.006606
Situação 5	0:35:53	0.007489922	0.006190326	0.005787

Em análise às tabelas 24 e 25, é possível observar que a duração do treino no cenário 2 é mais do dobro do tempo do cenário 1, devido às limitações computacionais dos Raspberry Pi's, evidenciado também nos modelos anteriores. Relativamente à perda, comparando o cenário 1 com o cenário 2, ambos apresentaram 3 melhores resultados. Com base nos resultados das simulações, é possível observar que a configuração **'return_sequences = False'** reduz a duração do treino dos modelos, bem como a perda na maioria dos casos. O melhor resultado, em relação à perda foi na "Situação 1" no cenário 2. Para o cenário 1 serão apresentados os resultados referentes à "Situação 4", dado que a perda mais baixa nesse cenário já foi estudada na secção 5.4.4. Os resultados referentes à "Situação 3", também serão apresentados. A figura 60, representa a perda tanto no treino como na validação, para a "Situação 1" no cenário 2 e para a "Situação 4" no cenário 1, enquanto a figura 61 é referente à "Situação 3" para cada cenário.



(a)

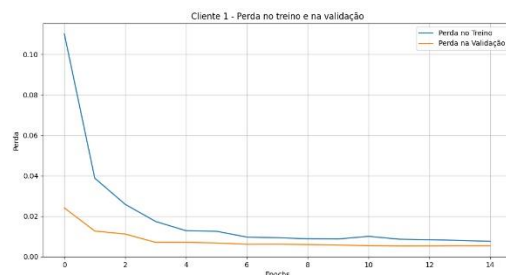


(b)

Figura 60 - Perda no treino e na validação do modelo GRU: (a) Cenário 1 – Situação 4; (b) Cenário 2 – Situação 1



(a)



(b)

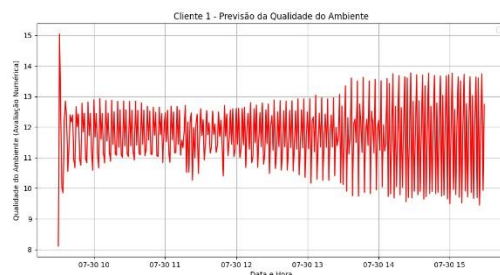
Figura 61 - Perda no treino e validação do modelo GRU na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário

2

Em análise às figuras 60 e 61, a perda obtida na “Situação 1” foi a mais baixa entre todas as amostragens, indicando que o parâmetro **‘return_sequences = False’** foi uma escolha acertada para este modelo. Por outro lado, na “Situação 3”, a perda obtida foi uma das mais elevadas entre as diferentes situações referentes aos dois cenários, não sendo das mais altas entre todas as amostragens. Para compreender como a perda afeta o desempenho destes modelos, seguem-se as previsões para cada situação e cenário, representados nas figuras 62 e 63.

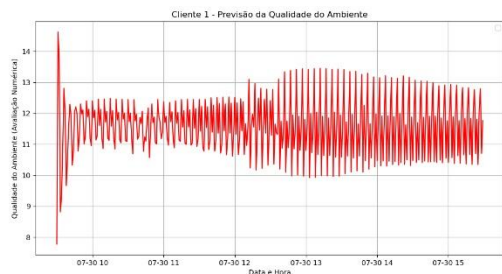


(a)

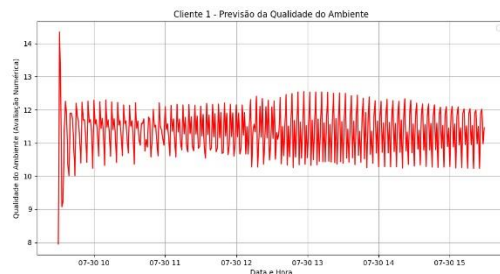


(b)

Figura 62 - Previsão dada pelo modelo GRU: (a) Cenário 1 – Situação 4; (b) Cenário 2 – Situação 1



(a)



(b)

Figura 63 - Previsão dada pelo modelo GRU na Situação 3 para cada cenário: (a) Cenário 1; (b) Cenário 2

Em análise às figuras 62 e 63, observa-se que os resultados da previsão obtidos nas situações 1 e 4 foram superiores aos da “Situação 3” até aproximadamente às 15h. Após essa hora, o comportamento da previsão da “Situação 3” foi bastante semelhante ao dos dados reais. Tanto a previsão da “Situação 1” como da “Situação 4”, foi bastante satisfatória, à exceção dos picos no início da previsão. Em ambos os cenários, os valores previstos estão dentro das gamas esperadas. Embora os resultados da “Situação 3”, tenham sido aceitáveis, eles foram inferiores às previsões das outras situações, sendo que nenhuma delas foi perfeita.

Com base nos resultados das simulações, observou-se que a configuração **'return_sequences = False'** reduz a duração do treino dos modelos. Além disso, esta configuração pode nunca levar a um aumento significativo da função de perda. Apenas em 5 situações de 12 possíveis, este parâmetro aumentou a perda, sendo que os aumentos nunca foram muito significativos. Para modelos que lidam com séries temporais, configurar **'return_sequences = False'** pode não ser a abordagem mais adequada. Quando **'return_sequences'** está definido como **('False')**, a camada recorrente retorna apenas o último output da sequência, em vez de toda a sequência de outputs. Isso simplifica o processamento da rede neuronal e reduz o tempo de treino. Nas simulações, verificou-se que os modelos com **'return_sequences = False'** completaram o treino mais rapidamente em comparação com aqueles configurados com **'return_sequences = True'**. Apesar da vantagem em termos de redução do tempo de treino, esta configuração pode resultar num aumento da função de perda em certos modelos. Isto acontece porque, ao retornar apenas a última saída, a rede pode perder informações importantes das etapas intermédias da sequência. Como resultado, a capacidade do modelo para captar padrões temporais complexos é reduzida, aumentando a função de perda. Para modelos que usam séries temporais, retornar apenas a última saída **'return_sequences = False'** pode não ser a melhor escolha, uma vez que séries temporais frequentemente contêm padrões que se desenvolvem ao longo do tempo. Configurar **'return_sequences = True'** permite que o modelo capture essas variações temporais ao longo da sequência, melhorando a precisão das previsões. Muitas séries temporais apresentam dependências de longo prazo, onde o valor atual é influenciado por muitos passos anteriores. Nesse contexto, **'return_sequences = True'** ajuda o modelo a manter essas dependências ao longo da sequência, proporcionando melhores resultados. No entanto, nem sempre foram atingidos os melhores resultados com **'return_sequences = True'**, sendo que em 7 situações de 12 possíveis, foram apresentados os melhores resultados com **'return_sequences = False'**. Notar que cada caso é um caso, e que para este caso o **'return_sequences = False'** foi a escolha certa para este tipo de modelos e dados. Isto porque, a variabilidade inerente ao processo de treino pode levar a resultados distintos entre diferentes

execuções do modelo. A escolha do modelo e dos parâmetros, terá de ter em conta a duração, a perda bem como a previsão. Sendo que o modelo RNN aparenta ser o melhor dado que, apresenta a menor duração do treino de todos os modelos e apresenta previsões bastante consistentes, especialmente para uma amostragem de 10 minutos.

CONCLUSÃO

O estudo presente nesta dissertação foi motivado pela necessidade de otimizar as condições de trabalho em edifícios de escritórios, utilizando uma abordagem baseada em *Federated Learning* (FL). O objetivo principal passa por melhorar a qualidade do ambiente, tendo em conta diversos fatores como a temperatura, qualidade do ar, ruído, iluminação e humidade.

Foi então desenvolvida uma solução que integra três Raspberry Pi 3, referentes aos dispositivos B-Health responsáveis pela recolha dos dados, 3 Raspberry Pi 4 para o treino dos modelos de regressão e *Machine Learning* (ML), e um servidor central, responsável pela agregação dos parâmetros dos modelos. O estudo comparou diversos algoritmos, incluindo algoritmos de regressão como ExtraTrees, Random Forest, xGBoost e Gradient Boosting, um modelo específico de séries temporais Facebook Prophet, bem como modelos de *Deep Learning* como RNN, GRU e LSTM.

Os resultados obtidos foram promissores. Os modelos de regressão, especialmente xGBoost e Gradient Boosting, apresentaram excelentes resultados em termos da função de perda e previsões consistentes, especialmente para uma amostragem de 1 minuto. Contudo, as previsões apresentadas nem sempre correspondiam com os dados reais. O Facebook Prophet teve um desempenho inferior quando comparado com os modelos de regressão, contudo não foi integrado com FL, devido à sua incompatibilidade com a *framework*, o que pode ter prejudicado o seu desempenho. Entre os modelos de *Deep Learning* testados, os melhores resultados, no geral ocorreram para uma amostragem de 10 minutos. Dos 3 modelos simulados, o LSTM foi o que apresentou os piores resultados tanto na perda como na previsão, tendo alguns pontos positivos a seu favor, principalmente no início da sua previsão dado que a subida dos valores previstos se assemelha à dos dados reais. Quanto aos outros dois modelos, RNN e GRU, ambos apresentaram resultados bastante interessantes, com destaque para o RNN dado que a sua duração do treino é bastante inferior à dos outros modelos.

Um aspeto importante observado foi o impacto da configuração `'return_sequences'` nos modelos de *Deep Learning*. Curiosamente, apesar de `'return_sequences=True'`, ser teoricamente mais adequado para séries temporais, os melhores resultados foram frequentemente obtidos com `'return_sequences=False'` neste estudo específico. Notar que nem sempre resultou

em melhores resultados, sendo necessária sempre uma análise cuidadosa e testes extensivos ao implementar este tipo de soluções para problemas do dia a dia.

Apesar dos resultados promissores, é importante destacar algumas limitações encontradas durante o estudo, que também apontam para aspectos cruciais a serem melhorados em pesquisas futuras. O problema encontrado quanto à descalibração do sensor da qualidade do ar referente ao cliente 1, comprometendo a qualidade tanto dos dados reais como da previsão. Além disso, os resultados do teste de estacionariedade, revelaram que este conjunto de dados referente à qualidade do ar do cliente 1 apresentava características de uma série não estacionária. Contudo, os modelos usados para as simulações têm a capacidade de trabalhar com séries não estacionárias. Principalmente, o RNN, LSTM e GRU que são projetados para detectar dependências temporais complexas.

Concluindo, apesar das limitações encontradas, este estudo demonstrou que a abordagem baseada em FL pode ser eficazmente aplicada para otimizar as condições de em edifícios de escritórios. Os modelos de DL, particularmente RNN e GRU, mostraram-se promissores neste contexto, oferecendo um bom equilíbrio entre precisão e eficiência computacional. A abordagem federada demonstrou potencial para melhorar significativamente as condições de trabalho.

Para desenvolvimentos futuros seria interessante, para além da melhoria das técnicas de pré-processamento de dados e afinação dos modelos em estudo, a implementação de um sistema de recomendações em tempo real baseado na qualidade do ambiente. As recomendações podem incluir ajustes na iluminação, ventilação, temperatura, entre outros. Outro exemplo, poderia ser uma integração do sistema FL com os sistemas de automação predial existentes, como o HVAC.

BIBLIOGRAFIA

- [1] Bureau of Labor Statistics and U.S Department of Labor, "Occupational Outlook Handbook." Accessed: Nov. 10, 2023. [Online]. Available: <https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm>
- [2] "REGULAMENTO (UE) 2016/ 679 DO PARLAMENTO EUROPEU E DO CONSELHO - de 27 de abril de 2016 - relativo à proteção das pessoas singulares no que diz respeito ao tratamento de dados pessoais e à livre circulação desses dados e que revoga a Diretiva 95/ 46/ CE (Regulamento Geral sobre a Proteção de Dados)," May 2016.
- [3] M. Crabtree, "What is Machine Learning? Definition, Types, Tools & More," Datacamp. Accessed: Nov. 10, 2023. [Online]. Available: <https://www.datacamp.com/blog/what-is-machine-learning>
- [4] D. Scalerandi, "Pros and Cons of Decentralized Network Applications," BairesDevBlog. Accessed: Dec. 29, 2023. [Online]. Available: <https://www.bairesdev.com/blog/pros-and-cons-of-decentralized-network-apps/>
- [5] Polaris Market Research, "Federated Learning Market Size Global Report, 2022 - 2030," Sep. 2022. Accessed: Dec. 09, 2023. [Online]. Available: <https://www.polarismarket-research.com/industry-analysis/federated-learning-market>
- [6] kbv research, "Global Federated Learning Market Report," May 2022. Accessed: Dec. 09, 2023. [Online]. Available: <https://www.kbvresearch.com/federated-learning-market/>
- [7] IEEE Xplore, "Documentation regarding Federated Learning," IEEE Xplore. Accessed: Dec. 12, 2023. [Online]. Available: <https://ieeexplore.ieee.org/search/searchresult.jsp?queryText=federated%20learning&highlight=true&returnType=SEARCH&matchPubs=true&returnFacets=ALL>
- [8] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *BULLETIN OF MATHEMATICAL BIOPHYSICS*, vol. 5, pp. 115–133, 1943, doi: 10.1007/BF02478259.

- [9] A. M. Turing, "MIND A QUARTERLY REVIEW OF PSYCHOLOGY AND PHILOSOPHY I.-COMPUTING MACHINERY AND INTELLIGENCE," 1950. [Online]. Available: <https://academic.oup.com/mind/article/LIX/236/433/986238>
- [10] J. Howard, "Artificial intelligence: Implications for the future of work," *Am J Ind Med*, vol. 62, no. 11, pp. 917–926, Nov. 2019, doi: 10.1002/ajim.23037.
- [11] F. Rosenblatt, "THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN 1," *Psychol Rev*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.
- [12] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal*, vol. 3, no. 3, pp. 211–229, Jul. 1959, doi: 10.1147/rd.33.0211.
- [13] Queensland Brain Institute, "History of Artificial Intelligence," Queensland Brain Institute. Accessed: Dec. 13, 2023. [Online]. Available: <https://qbi.uq.edu.au/brain/intelligent-machines/history-artificial-intelligence>
- [14] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," Dec. 2019, [Online]. Available: <http://arxiv.org/abs/1912.04977>
- [15] Blockforce, "5 Problems with centralized Data," Medium. Accessed: Nov. 12, 2023. [Online]. Available: <https://blockforce-in.medium.com/5-problems-with-centralized-data-e58deb1687b1>
- [16] M. Asad, A. Moustafa, and T. Ito, "Federated Learning Versus Classical Machine Learning: A Convergence Comparison," Nagoya, 2020.
- [17] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach," *IEEE Internet Things J*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020, doi: 10.1109/JIOT.2020.2991401.
- [18] H. Hosseini, S. Yun, H. Park, C. Louizos, J. Soriaga, and M. Welling, "Federated Learning of User Authentication Models," Jul. 2020, [Online]. Available: <http://arxiv.org/abs/2007.04618>
- [19] V. D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient Federated Learning Algorithm for Resource Allocation in Wireless IoT Networks," *IEEE*

- Internet Things J*, vol. 8, no. 5, pp. 3394–3409, Mar. 2021, doi: 10.1109/JIOT.2020.3022534.
- [20] H. Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise Agüera-Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” 2017.
- [21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated Learning: Challenges, Methods, and Future Directions,” *IEEE Signal Process Mag*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.
- [22] M. Asad *et al.*, “Limitations and Future Aspects of Communication Costs in Federated Learning: A Survey,” Sep. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/s23177358.
- [23] J. Kim, T. Park, H. Kim, and S. Kim, “Federated Learning for Face Recognition,” in *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021. doi: 10.1109/ICCE50685.2021.9427748.
- [24] M. Jansson and M. Axelsson, “Federated Learning Used to Detect Credit Card Fraud,” LUND University, 2020.
- [25] Z. Su *et al.*, “Secure and Efficient Federated Learning for Smart Grid with Edge-Cloud Collaboration,” *IEEE Trans Industr Inform*, vol. 18, no. 2, pp. 1333–1344, Feb. 2022, doi: 10.1109/TII.2021.3095506.
- [26] D. Jallepalli, N. C. Ravikumar, P. V. Badarinath, S. Uchil, and M. A. Suresh, “Federated Learning for Object Detection in Autonomous Vehicles,” in *Proceedings - IEEE 7th International Conference on Big Data Computing Service and Applications, BigDataService 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 107–114. doi: 10.1109/BigDataService52369.2021.00018.
- [27] N. Victor *et al.*, “Federated Learning for IoUT: Concepts, Applications, Challenges and Opportunities,” Jul. 2022, [Online]. Available: <http://arxiv.org/abs/2207.13976>
- [28] Q. Li *et al.*, “A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection,” *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 35, no. 4, pp. 3347–3366, Jul. 2019, doi: 10.1109/TKDE.2021.3124599.

- [29] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans Intell Syst Technol*, vol. 10, no. 2, Jan. 2019, doi: 10.1145/3298981.
- [30] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," Jul. 01, 2021, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/COMST.2021.3075439.
- [31] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A Secure Federated Transfer Learning Framework," *IEEE Intell Syst*, vol. 35, no. 4, pp. 70–82, Jul. 2020, doi: 10.1109/MIS.2020.2988525.
- [32] M. Alazab, S. P. Rm, M. Parimala, P. K. R. Maddikunta, T. R. Gadekallu, and Q. V. Pham, "Federated Learning for Cybersecurity: Concepts, Challenges, and Future Directions," *IEEE Trans Industr Inform*, vol. 18, no. 5, pp. 3501–3509, May 2022, doi: 10.1109/TII.2021.3119038.
- [33] A. Yigit Ogun, "Byzantine Generals Problem," Medium. Accessed: Dec. 11, 2023. [Online]. Available: <https://medium.com/@ayogun/byzantine-generals-problem-a47b33ef87fc>
- [34] "Model Inversion," Nightfall AI. Accessed: Dec. 12, 2023. [Online]. Available: <https://www.nightfall.ai/ai-security-101/model-inversion>
- [35] A. S.Gillis, "What is homomorphic encryption?," TechTarget. Accessed: Dec. 11, 2023. [Online]. Available: <https://www.techtarget.com/searchsecurity/definition/homomorphic-encryption>
- [36] IEEE, "What is Differential Privacy?," IEEE Digital Privacy. Accessed: Dec. 11, 2023. [Online]. Available: <https://digitalprivacy.ieee.org/publications/topics/what-is-differential-privacy>
- [37] J. Sen, "Peer-to-peer networks," in *National Conference on Emerging Trends in Advanced Computer Science*, Kolkata: IEEE, Mar. 2012. doi: 10.1109/NCETACS.2012.6203284.

- [38] O. Marfoq, C. Xu, G. Neglia, and R. Vidal, "Throughput-Optimal Topology Design for Cross-Silo Federated Learning," Oct. 2020, [Online]. Available: <http://arxiv.org/abs/2010.12229>
- [39] C. Huang, J. Huang, and X. Liu, "Cross-Silo Federated Learning: Challenges and Opportunities," Jun. 2022, [Online]. Available: <http://arxiv.org/abs/2206.12949>
- [40] "Time Series Analysis Explained," Sigma Computing. Accessed: Jul. 10, 2024. [Online]. Available: <https://www.sigmacomputing.com/resources/learn/what-is-time-series-analysis>
- [41] A. Singh, "Multivariate Time Series Analysis," Analytics Vidhya. Accessed: Jun. 20, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/>
- [42] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of Things (IoT): A Literature Review," *Journal of Computer and Communications*, vol. 03, no. 05, pp. 164–173, 2015, doi: 10.4236/jcc.2015.35021.
- [43] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, and F. Piccialli, "Model aggregation techniques in federated learning: A comprehensive survey," *Future Generation Computer Systems*, vol. 150, pp. 272–293, Jan. 2024, doi: 10.1016/j.future.2023.09.008.
- [44] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, May 2022, doi: 10.1109/iotm.004.2100182.
- [45] A. Ghosh, O. Khalid, R. N. B. Rais, A. Rehman, S. U. R. Malik, and I. A. Khan, "Data offloading in IoT environments: modeling, analysis, and verification," *EURASIP J Wirel Commun Netw*, vol. 2019, no. 1, Dec. 2019, doi: 10.1186/s13638-019-1358-8.
- [46] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet Things J*, vol. 9, no. 1, pp. 1–24, Jan. 2022, doi: 10.1109/JIOT.2021.3095077.
- [47] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," Dec. 2018, [Online]. Available: <http://arxiv.org/abs/1812.06127>
- [48] S. Reddi *et al.*, "Adaptive Federated Optimization," Feb. 2020, [Online]. Available: <http://arxiv.org/abs/2003.00295>

- [49] T. Toharudin, R. S. Pontoh, R. E. Caraka, S. Zahroh, Y. Lee, and R. C. Chen, "Employing long short-term memory and Facebook prophet model in air temperature forecasting," *Commun Stat Simul Comput*, vol. 52, no. 2, pp. 279–290, 2023, doi: 10.1080/03610918.2020.1854302.
- [50] Simplilearn, "What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning," Simplilearn. Accessed: Sep. 13, 2024. [Online]. Available: <https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article>
- [51] S. R, "Understanding Random Forest Algorithm With Examples," Analytics Vidhya. Accessed: Aug. 07, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>
- [52] B. Tsuchiev, "A Guide to the Gradient Boosting Algorithm," Datacamp. Accessed: Sep. 13, 2024. [Online]. Available: <https://www.datacamp.com/tutorial/guide-to-the-gradient-boosting-algorithm>
- [53] "How Extra trees classification and regression algorithm works," ArcGIS. Accessed: Sep. 13, 2024. [Online]. Available: <https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-extra-tree-classification-and-regression-works.htm>
- [54] Shiri F, Perumal T, Mustapha N, and Mohamed R, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU," 2023. Accessed: Aug. 06, 2024. [Online]. Available: <https://arxiv.org/abs/2305.17473>
- [55] S. Githu, "A Federated Approach to Train and Deploy Machine Learning Models," Edge Impulse. Accessed: Jan. 15, 2024. [Online]. Available: <https://docs.edgeimpulse.com/experts/software-integration-demos/federated-learning-raspberry-pi#introduction>
- [56] J. Fernandez-Marques, "Running Federated Learning applications on Embedded Devices," Flower. Accessed: Jan. 15, 2024. [Online]. Available: https://flower.dev/blog/2020-12-16-running_federated_learning_applications_on_embedded_devices_with_flower/

- [57] I. Varlamis *et al.*, “Using big data and federated learning for generating energy efficiency recommendations,” *Int J Data Sci Anal*, vol. 16, no. 3, pp. 353–369, Sep. 2023, doi: 10.1007/s41060-022-00331-2.
- [58] L. Ridolfi, D. Naseh, S. S. Shinde, and D. Tarchi, “Implementation and Evaluation of a Federated Learning Framework on Raspberry PI Platforms for IoT 6G Applications,” *Future Internet*, vol. 15, no. 11, Nov. 2023, doi: 10.3390/fi15110358.
- [59] “Smart4Health.” Accessed: Jan. 27, 2024. [Online]. Available: <https://smart4health.eu/>
- [60] “ICU4COVID.” Accessed: Jan. 27, 2024. [Online]. Available: <https://www.icu4covid.eu/>
- [61] D. J. Beutel *et al.*, “Flower: A Friendly Federated Learning Research Framework,” Jul. 2020, [Online]. Available: <http://arxiv.org/abs/2007.14390>
- [62] A. Braungardt, “Flower, FATE, PySyft & Co. - Federated Learning Frameworks in Python,” Medium. Accessed: Sep. 11, 2024. [Online]. Available: <https://medium.com/elca-it/flower-pysyft-co-federated-learning-frameworks-in-python-b1a8eda68b0d>
- [63] Flower Framework, “Differential Privacy,” Flower Framework. Accessed: Sep. 11, 2024. [Online]. Available: <https://flower.ai/docs/framework/explanation-differential-privacy.html>



