Aprendizado Federado baseado em Múltiplas Árvores de Decisão para Aplicações IoT com Computação de Borda Cooperativa

Lucas Nobre Barbosa[1], **André Riker**[1]

[1]Faculdade de Computação – Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)

Abstract. *Internet of Things (IoT) have relied on edge computing nodes to decentralize computation and to bring more processing power near the IoT devices, such as sensors and actuators. IoT edge computing nodes have more data processing power and energy resources than regular IoT devices that aim to monitor and actuate on the environment. However, in general, IoT edge computing nodes are not designed for intensive Machine Learning (ML) training or to host large ML models. In the current IoT network architectures, there are multiple IoT edge computing nodes strategically located near a large number of IoT devices, where each of the IoT edge computing node has access to part of the data produced by the whole IoT network. In this scenario, each IoT edge computing node runs lightweight ML models in its local dataset. In this paper, we propose a solution, called FEDT (FEderated Decision Tree), that aggregates the learning produced by multiple decision trees from cooperative IoT edge nodes, following the federated learning principles. We present four different federated learning strategies and demonstrate that FEDT can achieve around 80% of a centralized ML model in terms of Pearson correlation.*

Resumo. *A Internet das Coisas (IoT) tem dependido de nós de computação em borda para descentralizar a computação e trazer mais poder de processamento próximo aos dispositivos IoT, como sensores e atuadores. Os nós de computação de borda da IoT têm mais poder de processamento de dados e recursos energéticos do que os dispositivos IoT regulares que visam monitorar e atuar no ambiente. No entanto, em geral, os nós de computação de borda não são projetados para treinamento intensivo de Aprendizado de Máquina (ML) ou para hospedar grandes modelos de ML. Nas arquiteturas de rede IoT atuais, existem múltiplos nós de computação de borda estrategicamente localizados perto de um grande número de dispositivos, onde cada um dos nós de computação de borda tem acesso a parte dos dados produzidos por toda a rede IoT. Neste cenário, cada nó de computação de borda executa modelos de ML leves em seu conjunto de dados local. Neste artigo, propomos uma solução, chamada FEderated Decison Tree (FEDT), Árvore de Decisão Federada, que agrega o aprendizado produzido por múltiplas árvores de decisão de nós de borda cooperativos, seguindo os princípios de aprendizado federado. Apresentamos quatro estratégias de aprendizado federado diferentes e demonstramos que o FEDT pode alcançar cerca de 80% de um modelo de ML centralizado em termos de correlação de Pearson.*

1. Introdução

Por mais de uma década, a Internet das Coisas, do Inglês *Internet of Things* (IoT), tem atraído grande atenção, pois conecta um número massivo de dispositivos e possibilita uma ampla variedade de aplicações. Recentemente, a IoT fundiu-se com a computação de borda para fornecer serviços mais rápidos e inteligentes, já que a computação de borda traz algumas das capacidades de nuvem para perto dos dispositivos de consumo de computação [Shen et al. 2022]. Uma arquitetura de rede orientada a reunir IoT e computação de borda depende de um conjunto de dispositivos IoT para fornecer coleta de dados, sensoriamento, atuação e comunicação, e um conjunto de nós de borda cooperativos para suportar tarefas intensivas de computação, como tarefas de aprendizado de máquina, para os dispositivos IoT.

Prover privacidade é uma necessidade urgente para quase todas as aplicações IoT, seja para cumprir a legislação ou devido à desconfiança dos usuários sobre como seus dados sensíveis podem ser usados. O Aprendizado Federado, do Inglês *Federated Learning* (FL) emerge como uma solução para esse problema, pois é uma abordagem de Aprendizado de Máquina, em Inglês *Machine Learning* (ML) distribuída capaz de fornecer um nível extra de proteção da privacidade dos dados do usuário [Huang et al. 2023]. No FL, vários modelos de ML são executados em conjuntos de dados locais sensíveis à privacidade, simultaneamente, e um modelo de ML global, em execução em um servidor, é construído sem compartilhar os conjuntos de dados locais com o servidor.

Neste artigo, consideramos um cenário em que múltiplos nós de borda estão dispersos por uma rede IoT, fornecendo serviços de computação cooperativa para subconjuntos da rede. Por exemplo, uma área residencial é coberta por uma rede IoT, e cada residência é um subconjunto da rede assistido por um nó de borda. Neste cenário, chamado de Borda Cooperativa para IoT Residencial (CERIoT), o nó de borda pode ter acesso aos dados de uma única residência. O dispositivo de borda pode criar um conjunto de dados local com os dados coletados dos dispositivos IoT implantados em uma residência. Compartilhar os dados de uma residência com um serviço de terceiros, como computação em nuvem, pode ser negado pelo usuário, pois gera preocupações em termos de privacidade dos dados. Acreditamos que o FL é uma abordagem adequada para este cenário, pois preserva a privacidade dos dados locais dos moradores, mas também permite serviços baseados em ML para aplicações inteligentes.

A maioria das soluções de ponta basedas em FL, por exemplo, [Ji and Chen 2023] [Yu and Li 2021], são projetadas para usar modelos de ML executando o algoritmo Stochastic Gradient Descent (SGD), como FedAVG [Reddi et al. 2020]. A literatura possui muitas variantes de soluções FL baseadas em SGD. Uma das poucas soluções não baseadas em SGD, Shen et al. [Shen et al. 2022] propõem uma solução de ML distribuída projetada com floresta aleatória. O foco desta solução é determinar como as árvores de decisão serão distribuídas sobre os nós de borda no cenário CERIoT. No entanto, a solução proposta por [Shen et al. 2022] não é projetada de acordo com os princípios do FL, pois os autores não mostram como agregar as árvores locais treinadas em um único modelo de ML global.

Neste artigo, a solução proposta, chamada FEDT (Árvore de Decisão Federada), é uma solução baseada em princípios do FL que é projetada especificamente para Florestas Aleatórias e Árvores de Decisão. A razão para projetar uma solução FL usando Árvores

de Decisão no cenário CERIoT é dupla: (i) A literatura é extensa sobre este tópico, dando suporte para Árvores de Decisão em conjuntos de dados tabulares em pequena escala [Grinsztajn et al. 2022] [Lindskog and Prehofer 2023]; (ii) O custo de treinamento das Árvores de Decisão é conhecido por ser pequeno quando comparado com outros modelos de ML, como ANN e SVM [Maseer et al. 2021]. Portanto, propomos e avaliamos uma estratégia FL a ser executada nos clientes FEDT e quatro estratégias para o servidor FL produzir um modelo global baseado em Árvore de Decisão. Os testes de desempenho de todas as estratégias propostas foram conduzidos usando um conjunto de dados [Candanedo et al. 2017] que contém o consumo de energia de eletrodomésticos de 14.803 residências.

Este artigo está organizado da seguinte forma. A Seção 2 aborda a base teórica para compreender a solução proposta. A Seção 3 apresenta diversas abordagens para problemas de Aprendizado Federado. A Seção 4 apresenta as soluções propostas e as estratégias FL. A Seção 5 detalha a avaliação de desempenho e os resultados obtidos. A Seção 6 apresenta a conclusão e os trabalhos futuros.

2. Referencial Teórico

2.1. Árvores de Decisão

As árvores de decisão são modelos de aprendizado de máquina amplamente utilizados devido à sua simplicidade e interpretabilidade. Elas são estruturas em forma de árvore onde cada nó interno representa uma decisão com base em um atributo dos dados, e cada folha representa uma classe ou valor de saída. A construção da árvore de decisão envolve a seleção dos atributos mais importantes para dividir os dados em subgrupos homogêneos, de modo que a árvore seja capaz de fazer previsões precisas.

Embora as árvores de decisão sejam eficazes em muitos cenários, elas também têm suas limitações. Por exemplo, elas podem ser sensíveis a pequenas variações nos dados de treinamento e propensas a overfitting quando não são regularizadas adequadamente. No entanto, essas limitações podem ser mitigadas por meio de técnicas avançadas, como a poda da árvore e o uso de ensemble methods, como as florestas aleatórias.

2.2. Florestas Aleatórias

As florestas aleatórias são uma extensão das árvores de decisão que visam melhorar a precisão e a estabilidade do modelo. Em vez de construir uma única árvore de decisão, as florestas aleatórias criam múltiplas árvores em paralelo, cada uma treinada em uma amostra aleatória dos dados e com um subconjunto aleatório dos atributos. A previsão final é obtida por meio da votação ou média das previsões de todas as árvores na floresta.

Uma das principais vantagens das florestas aleatórias é sua capacidade de lidar com dados heterogêneos e complexos, além de reduzir o overfitting em comparação com árvores de decisão individuais. Além disso, as florestas aleatórias são robustas a outliers e ruído nos dados, o que as torna uma escolha popular para uma variedade de problemas de aprendizado de máquina.

3. Trabalhos Relacionados

[Yu and Li 2021] investiga a aplicação de técnicas de otimização de recursos em aprendizado federado (FL) em computação móvel de borda, reconhecendo os desafios relaciona-

dos à intensidade dos recursos dos dispositivos móveis em termos de computação, largura de banda, energia e dados.

[Candanedo et al. 2017] aborda a predição de consumo energético de dispositivos por meio de técnicas de aprendizado centralizado. Seu trabalho se concentra em explorar modelos de previsão de consumo energético com base em dados coletados dos dispositivos. Esta pesquisa contribui para o entendimento das capacidades e limitações desses métodos em relação à predição precisa do consumo de energia em dispositivos.

[Shen et al. 2022] investiga abordagens distribuídas para a predição de consumo de energia, explorando as vantagens e desvantagens de cada uma. Essa abordagem proporciona uma visão abrangente das estratégias distribuídas disponíveis para prever o consumo energético, permitindo uma compreensão mais completa das complexidades envolvidas nesse contexto.

[Hauschild et al. 2022] investiga a eficácia dos modelos de Random Forests federados (FRF) na medicina de precisão, destacando sua capacidade de superar modelos locais em cenários com conjuntos de dados heterogêneos e desequilibrados. Os resultados demonstram que os FRF oferecem uma abordagem promissora para colaborações clínicas sem violar a privacidade dos dados, permitindo a construção de modelos mais generalizáveis para melhorar as decisões clínicas e facilitar o tratamento personalizado em contextos de saúde diversificados.

[Markovic et al. 2022] propõe uma abordagem de aprendizado federado para detecção de intrusões, visando mitigar a vulnerabilidade dos dados ao treinar modelos de Random Forest (RF) de forma distribuída em múltiplos clientes. Os resultados mostram que o RF global no servidor alcança maior precisão do que os RFs individuais nos clientes em dois dos quatro conjuntos de dados avaliados, com desempenho próximo ao máximo em um terceiro conjunto de dados. Mesmo no quarto caso, o RF global supera a precisão média, embora fique atrás do máximo.

[Huang et al. 2023] aborda a questão do aprendizado federado (FL) em um framework de aprendizado distribuído em larga escala com proteção de privacidade do usuário por meio de treinamento local e agregação global. O estudo propõe melhorar a taxa de convergência sob o sistema FL assíncrono com garantia de desempenho e quantização, considerando o equilíbrio entre o erro de quantização (QE) e a obsolescência. Os resultados demonstram a importância de encontrar o trade-off adequado entre a taxa de convergência e o QE sob um sistema FL assíncrono e quantizado.

[Reddi et al. 2020] propõe abordagens para otimização adaptativa em aprendizado federado (FL), visando melhorar o desempenho do processo de treinamento em ambientes com dados heterogêneos.

[Ji and Chen 2023] introduz o FedQNN, um framework de aprendizado federado (FL) eficiente em termos de computação e comunicação para cenários de IoT. Esta abordagem inova ao integrar a quantização de ultrabaixa largura de bits no ambiente FL, permitindo que os clientes realizem computação eficiente de ponto fixo com menos consumo de energia. Além disso, os dados tanto de ida quanto de volta são significativamente comprimidos para uma comunicação mais eficiente, utilizando uma combinação de estratégias de esparsificação e quantização.

[Singhal et al. 2024] desenvolve uma estratégia de seleção de clientes enviesada, chamada GREEDYFED, para melhorar a convergência rápida do treinamento de modelos em Aprendizado Federado (FL), especialmente em cenários práticos com heterogeneidade significativa nos recursos de dados, computação e comunicação entre os clientes. Esta abordagem baseia-se em um algoritmo de aproximação rápida para o Valor de Shapley no servidor de parâmetros, tornando o cálculo viável para aplicações do mundo real com muitos clientes. Os resultados mostram que o GREEDYFED demonstra convergência rápida e estável com alta precisão, mesmo sob restrições de tempo e com um alto grau de heterogeneidade nos dados, restrições de sistema e requisitos de privacidade.

4. Árvores de Decisão Federadas

Este artigo apresenta uma estratégia para agregar árvores de decisão no servidor e quatro estratégias diferentes a serem executadas nos clientes, ou seja, nós de borda. O restante desta Seção apresenta uma visão geral da solução proposta em 4.1. A estratégia de agregação executada pelo servidor e pelos clientes é introduzida em 4.2 e 4.3.

4.1. Visão Geral do FEDT

A Figura 1 fornece uma visão geral do FEDT (Árvore de Decisão Federada), representando o aprendizado federado em cinco etapas distintas. Na primeira etapa, o processo começa com a inicialização do modelo de aprendizado pelo servidor central, que então transfere esse modelo para os nós de borda.

Na segunda etapa, os clientes de aprendizado federado entram em ação, realizando a fase de treinamento com seus conjuntos de dados locais. Essa etapa permite que os modelos locais sejam ajustados de acordo com os dados específicos de cada cliente.

Na terceira etapa, os clientes enviam suas atualizações, ou seja, os parâmetros de aprendizado do modelo, de volta para o servidor central. Esse processo de comunicação garante que o servidor central esteja ciente das atualizações feitas em cada nó de borda.

Na quarta etapa, o servidor central assume a responsabilidade de agregar todas as atualizações recebidas dos clientes, combinando-as para formar um novo modelo global. Essa etapa garante a consistência e a precisão do modelo global.

Por fim, na quinta etapa, os parâmetros atualizados são transferidos de volta para os nós de borda, garantindo que todos os participantes do processo estejam sincronizados e prontos para iniciar um novo ciclo de treinamento. Essa abordagem em várias etapas do aprendizado federado auxilia a eficiência e eficácia do processo como um todo.

4.2. Estratégia Federada nos Nós de Borda

O pseudocódigo 1 apresenta como um cliente específico executa o FEDT. Primeiramente, ele inicializa o modelo local com informações enviadas pelo servidor. Em seguida, o cliente realiza o treinamento local com o conjunto de dados local, e o treinamento é realizado em rodadas, que são as interações entre os clientes e o servidor. Em cada rodada, o modelo local treinado tem um erro, ou seja, *localError*, que é computado usando o Erro Quadrático Médio (MSE). Da mesma forma, o modelo global recebido também tem um MSE. Caso o erro local tenha menos MSE do que o erro global, o modelo local é atribuído à variável de melhor modelo.

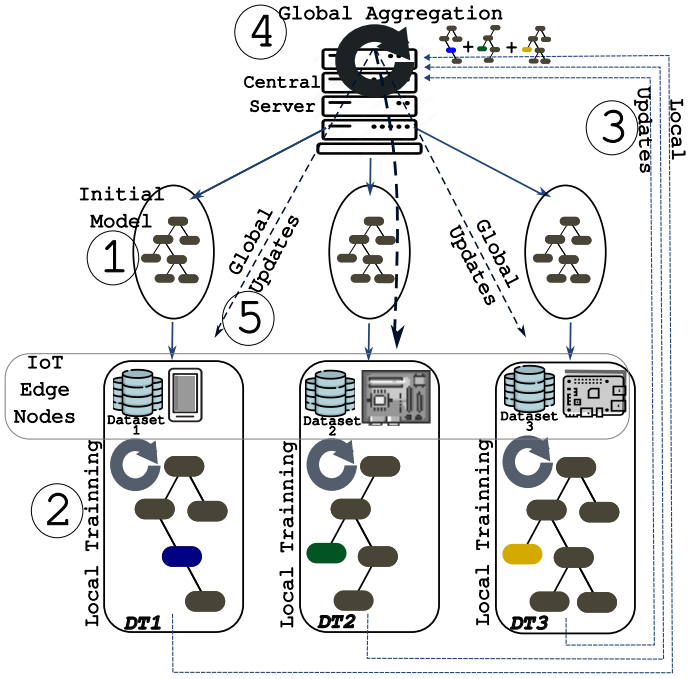

Figura 1. Arquitetura de Floresta Federada

É importante notar que, independentemente da estratégia do servidor FEDT, os clientes seguem esse mesmo processo de atualização. No final de cada rodada, o cliente envia para o servidor o melhor modelo atualizado.

Algorithm 1: FEDT Client

```
1  initializeLocalModel();
2  for each round r = 1, ...R do
3      receive globalTreeBasedModel;
4      train(localTreeBasedModel, localDataset)
         localError ← MSE(localTreeBasedModel);
5      globalError ← MSE(globalTreeBasedModel);
6      if globalError ≥ localError then
7          bestModel ← localTreeBasedModel;
8      else
9          if globalModelError < localModelError then
10             bestModel ← globalTreeBasedModel
11     Send to Server bestModel
```

4.3. Estratégias Federadas no Servidor

O pseudocódigo 2 apresenta como um servidor inicializa, agrega e atualiza os parâmetros vindos dos clientes, criando um modelo global.

Apesar de 2 introduzir quatro estratégias para agregar os modelos locais em um modelo global, a ideia principal por trás de todas as estratégias é: (i) após o servidor compartilhar o modelo global com todos os clientes, ele recebe os modelos locais atualizados, que são atualizados de acordo com o processo de treinamento realizado com o

conjunto de dados local nos clientes; (ii) o servidor seleciona os melhores modelos locais para agregá-los e compor o modelo global.

4.3.1. Estratégia 1: Melhores Árvores de Cada Cliente (MACC)

O servidor FL adota uma abordagem de seleção de modelos baseada em um percentual fixo, representado por *W%*, para determinar quais árvores de decisão enviadas pelos clientes serão selecionadas para compor o modelo global. Esse percentual fixo é escolhido com base em considerações práticas e pode variar dependendo das características específicas do problema e dos recursos disponíveis.

A escolha de um percentual fixo oferece uma maneira simples e eficaz de controlar a quantidade de modelos locais agregados ao modelo global, permitindo que o servidor mantenha um equilíbrio entre a diversidade e a qualidade dos modelos selecionados. Por exemplo, um percentual fixo mais baixo pode resultar em uma seleção mais restrita de modelos locais, favorecendo a precisão do modelo global, enquanto um percentual fixo mais alto pode promover uma maior diversidade de modelos locais, possibilitando uma melhor generalização do modelo global.

No entanto, é importante considerar que a escolha do percentual fixo ideal pode variar dependendo das características do conjunto de dados e dos objetivos do aprendizado federado. Portanto, é recomendável realizar experimentos empíricos para determinar o valor mais adequado de *W%* para cada aplicação específica.

4.3.2. Estratégia 2: Árvores Aleatórias de Cada Cliente (AACC)

O servidor FL adota uma abordagem de seleção aleatória entre todas as árvores de decisão dos clientes para compor o modelo global. Nessa abordagem, o servidor seleciona um percentual fixo, representado por *Z%*, das árvores de decisão dos clientes de forma aleatória. Essa seleção aleatória oferece uma maneira simples e eficiente de garantir que o modelo global contenha uma variedade representativa de modelos locais, contribuindo para uma melhor generalização e robustez do modelo global.

O percentual fixo *Z%* pode ser determinado com base em considerações práticas e pode variar dependendo das características específicas do problema e dos recursos disponíveis. Por exemplo, um valor mais baixo de *Z%* pode resultar em uma seleção mais restrita de modelos locais, favorecendo a precisão do modelo global, enquanto um valor mais alto de *Z%* pode promover uma maior diversidade de modelos locais, possibilitando uma melhor adaptação a diferentes distribuições de dados entre os clientes.

No entanto, é importante considerar que a escolha do valor ideal de *Z%* pode variar dependendo das características do conjunto de dados e dos objetivos do aprendizado federado. Portanto, é recomendável realizar experimentos empíricos para determinar o valor mais adequado de *Z%* para cada aplicação específica.

Algorithm 2: FEDT Server

```
Data: clientsModels, Strategy, W, Z, K
Result: globalModel
initializeGlobalModel();
initializeClients();
receive clientlModels();
switch Strategy do
    /* Best Trees from Every Client (BTEC)                    */
    case BTEC do
        while globalModel not in convergence do
            for trees in clientsModels do
                trees ← randomForest.trees;
                sortTreesByAccuracy();
                bestTrees ← trees.get(W%);
                globalModel.add(bestTrees);
            Send to Clients globalModel  Wait for clientsModels
    /* Random Trees from Every Client (RTEC)                  */
    case RTEC do
        while globalModel not in convergence do
            for trees in clientsModels do
                bestTrees ← randomForest.getRandom(Z%);
                globalModel.add(bestTrees);
            Send to Clients globalModel  Wait for clientsModels
    /* Best Forest (BF)                                       */
    case BF do
        while globalModel not in convergence do
            for Forests in clientsModels do
                forestError ← MSE(randomForest);
                sortForestsByAccuracy();
                bestForest ← getBestTree();
                globalModel.add(bestForest);
            Send to Clients globalModel  Wait for clientsModels
    /* Best Trees with Threshold Condition (BTTC)             */
    case BTTC do
        while globalModel not in convergence do
            for trees in clientsModels do
                trees ← randomForest.trees;
                bestTrees ← trees above K% accuracy;
                globalModel.add(bestTrees);
            Send to Clients globalModel  Wait for clientsModels
otherwise do
    Execute Default Strategy
```

4.3.3. Estratégia 3: Melhor Floresta (MF)

O servidor FL adota uma abordagem de seleção baseada na classificação por acurácia de todas as florestas aleatórias dos clientes para compor o modelo global. Nessa abordagem, o servidor avalia a acurácia de cada floresta aleatória enviada pelos clientes e classifica-as em ordem decrescente com base em suas pontuações de acurácia. Em seguida, o servidor seleciona a floresta aleatória com a melhor pontuação de acurácia para compor o modelo global.

No entanto, uma preocupação significativa com essa abordagem é a possível diminuição da diversidade do modelo resultante. A seleção baseada apenas na acurácia pode favorecer a escolha de modelos locais sem considerar outros aspectos importantes, como a representatividade dos dados ou a robustez do modelo. Isso pode levar a um modelo global menos diversificado, o que pode prejudicar sua capacidade de generalização e desempenho em diferentes cenários de aplicação.

Portanto, é importante considerar a diversidade do modelo global ao selecionar as florestas aleatórias dos clientes, buscando manter um equilíbrio entre a precisão e a diversidade para garantir a qualidade e a adaptabilidade do modelo resultante.

4.3.4. Estratégia 4: Melhores Árvores com Condição de Limiar (MACL)

Cada cliente contribui para o servidor FL enviando suas árvores de decisão. No servidor, uma estratégia é adotada para selecionar as árvores de decisão a serem adicionadas ao modelo global. Nessa estratégia, todas as árvores de decisão dos clientes que alcançam uma acurácia superior a um determinado limiar, representado por $K\%$, são consideradas para inclusão no modelo global.

Essa abordagem permite que apenas as árvores de decisão mais precisas contribuam para a construção do modelo global, garantindo assim a qualidade e o desempenho do modelo resultante. No entanto, é importante definir cuidadosamente o valor do limiar $K\%$ para equilibrar a precisão do modelo global com sua diversidade e capacidade de generalização. Um limiar muito alto pode resultar na exclusão de árvores de decisão úteis, enquanto um limiar muito baixo pode comprometer a qualidade do modelo global ao incluir árvores de decisão menos precisas.

Portanto, a escolha do valor ideal para o limiar $K\%$ deve ser baseada em considerações específicas do problema e em experimentos empíricos para avaliar seu impacto no desempenho do modelo global.

5. Avaliação de Desempenho e Resultados

Nesta seção, descrevemos a avaliação realizada e apresentamos os resultados obtidos para todas as estratégias de cliente federado comparadas com uma abordagem tradicional de floresta aleatória centralizada. Esta seção está organizada da seguinte forma: 5.1 descreve o conjunto de dados usado para realizar esta avaliação; 5.2 apresenta os detalhes sobre a implementação e as métricas selecionadas para medir o desempenho; 5.3 apresenta e discute os resultados obtidos.

5.1. Conjunto de Dados

Aplicamos as soluções propostas em dados de previsão de energia de eletrodomésticos [Candanedo et al. 2017]. Este conjunto de dados contém informações sobre quanta energia os eletrodomésticos de uma casa consomem em kilowatts hora. O mesmo considera diversos fatores ambientais como temperatura, umidade e velocidade do vento em diferentes áreas da residência, juntamente com dados de localização (obtidos de estações meteorológicas) coletados de 14.803 residências para treinamento e 4.932 para teste.

Este conjunto de dados reflete uma aplicação do mundo real para aprendizado federado. Primeiramente, as medidas dos sensores são coletadas em diferentes locais (por exemplo, casas), cada um com suas características únicas. Em segundo lugar, os proprietários não estão dispostos a compartilhar os dados dos sensores coletados de suas casas devido a considerações de privacidade.

Dividimos os dados em conjuntos de treinamento (80%) e teste (20%). Para imitar um ambiente distribuído, os dados de treinamento foram divididos ainda mais em vários pedaços (de 2 a 4 pedaços, cada um representando uma residência na mesma área). Isso garantiu diversidade entre as partições, refletindo condições do mundo real. Em seguida, aplicamos nossas políticas aos dados distribuídos e treinamos modelos de floresta aleatória.

5.2. Implementação e Métricas de Desempenho

A implementação do FEDT (Árvore de Decisão Federada) foi realizada utilizando a linguagem de programação Python, uma escolha comum devido à sua ampla gama de bibliotecas e facilidade de uso. Além disso, foram utilizadas duas bibliotecas principais: pandas, para o tratamento eficiente dos dados, e scikit-learn, uma biblioteca popular de aprendizado de máquina em Python, que oferece diversas ferramentas para modelagem de Inteligência Artificial.

Os códigos, que rodam em um ambiente WSL2 (*Windows Subsystem for Linux*), simulam um cenário de 4 casas com 1 dispositivo cliente cada, que realiza as computações com os dados de todos os sensores, e 1 servidor.

No processo de implementação, uma das principais decisões de design foi a escolha do modelo de aprendizado de máquina a ser utilizado. Optou-se por utilizar o RandomForestRegressor, uma variação do algoritmo Random Forest, que é altamente eficaz para problemas de regressão. Para garantir a capacidade do modelo de capturar relações complexas nos dados, foram definidos hiperparâmetros específicos, como mostrados na tabela 1.

Hiperparâmetro	Valor
Profundidade Máxima de uma Árvore	27
Número Máximo de Folhas em uma Árvore	6121
Estado Aleatório	42

Tabela 1. Hiperparâmetros definidos para cada Floresta Aleatória

Essa abordagem de implementação cuidadosamente selecionada foi fundamental para garantir que o FEDT fosse capaz de lidar eficientemente com conjuntos de dados

complexos e fornecer resultados precisos e confiáveis. Ao utilizar as bibliotecas pandas e scikit-learn em conjunto com o modelo RandomForestRegressor, o FEDT demonstrou ser uma solução robusta e escalável para problemas de aprendizado federado envolvendo árvores de decisão.

As seguintes métricas de desempenho foram consideradas para esta avaliação:

- **Correlação de Pearson:** O método de correlação de Pearson é um dos métodos mais comums para usar em variáveis numéricas; É uma abordagem baseada em similaridade que compara um objeto de dados com outro, atributo por atributo, geralmente somando os quadrados das diferenças de magnitude para cada atributo, e usando o cálculo para calcular um resultado final, conhecido como o escore de correlação [Berman 2016]. O resultado final é um valor entre -1 e 1, onde 0 é nenhuma correlação, 1 é correlação positiva total e -1 é correlação negativa total [Nettleton 2014]. Dados os dados em pares $(x_1, y_1), \ldots, (x_n, y_n)$ consistindo de n pares, a Correlação de Pearson r é definida como:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

- **Tempo de Treinamento:** Esta métrica mede o tempo, em segundos, a partir da inicialização do servidor e dos clientes FEDT e termina quando o modelo global atinge um determinado MSE.

5.3. Resultados Obtidos

A Figura 2 apresenta a correlação de Pearson obtida em cada rodada do processo de aprendizado federado. Durante esse processo, os modelos de florestas aleatórias são treinados em cada cliente com base em seus próprios conjuntos de dados locais. A correlação de Pearson é calculada para avaliar a consistência dos modelos locais em cada rodada de treinamento.

É importante observar que a abordagem centralizada, que tem acesso a todos os dados de treinamento do conjunto de dados, apresenta uma correlação de Pearson de 0,75. Para alcançar esse desempenho centralizado, foram testadas diferentes configurações de hiperparâmetros, como o número de árvores na floresta, a profundidade máxima das árvores e o número mínimo de amostras necessárias para dividir um nó.

No entanto, alcançar esse nível de desempenho centralizado no contexto federado é desafiador devido à distribuição heterogênea dos dados e à comunicação limitada entre os clientes e o servidor central. Portanto, o resultado apresentado é o melhor desempenho centralizado e federado que conseguimos alcançar após uma análise cuidadosa e extensiva dos parâmetros do modelo e das técnicas de treinamento.

Destaca-se que, para as estratégias que precisam de um valor definido para seu funcionamento, a tabela 2 demonstra os valores escolhidos para o ambiente de simulação.

Estratégia	Característica	Porcentagem Fixo
MACC	% de Árvores Selecionadas	50%
AACC	% de Árvores Selecionadas	50%
MACL	Pearson a ser superado	45%

Tabela 2. Porcentuais fixos definidos para cada Estratégia

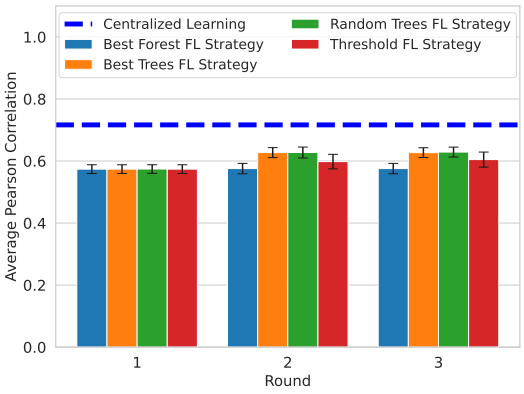

Figura 2. Correlação de Pearson por Round de Treinamento

A Figura 3 apresenta a correlação de Pearson obtida variando o número de árvores em cada cliente. Durante o experimento, o número mínimo de árvores para executar o FEDT foi definido como 10.

Para essa configuração inicial, observou-se que o desempenho variou entre 0,53 e 0,58, indicando uma correlação moderada entre os modelos locais e o modelo centralizado, que teve uma correlação de Pearson média de 0,72.

Conforme o número de árvores aumentava, o desempenho geral tendia a melhorar, atingindo valores próximos ou acima de 0,6 para um número suficientemente grande de árvores. No entanto, foi observado que, após cerca de 100 árvores, o ganho no desempenho tornou-se marginal, sugerindo uma possível saturação no benefício de adicionar mais árvores.

Esses resultados destacam a importância de encontrar um equilíbrio entre a complexidade do modelo e o desempenho, especialmente em cenários federados onde os recursos são limitados e a comunicação entre os clientes e o servidor é restrita.

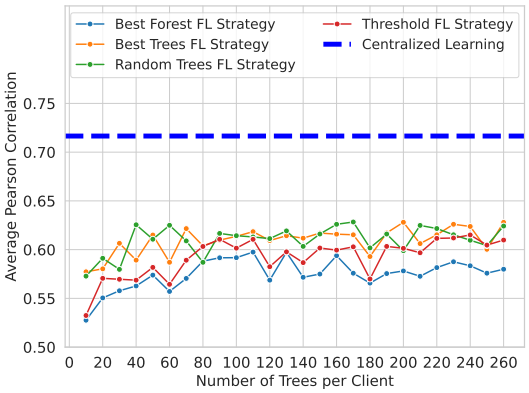

Figura 3. Correlação de Pearson por Número de Árvores em cada Cliente

As Figuras 4, 5 e 6 comparam o tempo de treinamento em segundos para cada uma das quatro estratégias apresentadas em relação a um dos três rounds de treinamento.

Cada figura representa esse tempo de treinamento para um intervalo de simulações com números de árvores diferentes entre elas.

A Figura 4 demonstra a média de tempo de treinamento com o intervalo que compreende as simulações que usaram de 10 a 100 árvores. A Figura 5 faz o mesmo para o intervalo de 110 a 200 árvores. Por fim, a 6 mostra os resultados para as simulações que usaram de 210 a 300 árvores durante o treinamento.

Os resultados demonstram a necessidade de equilibrar a complexidade do modelo com a eficiência computacional, onde a estratégia "Árvores Aleatórias de Cada Cliente (AACC)"apresenta a melhor relação entre Correlação de Pearson e Tempo de Treinamento.

Em conjunto, essas análises fornecem insights sobre o impacto do tempo de treinamento nas estratégias de cliente FEDT e orientam a seleção adequada de hiperparâmetros para otimizar o desempenho do sistema federado.

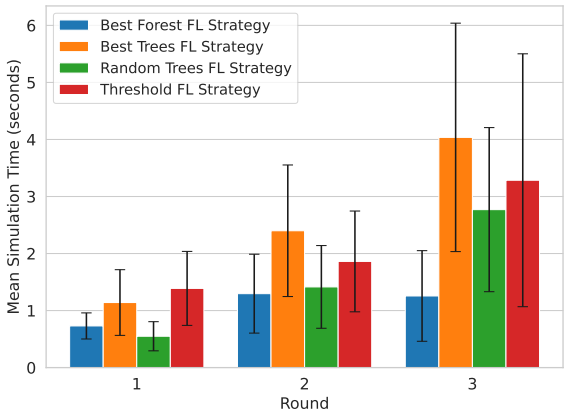

Figura 4. Tempo de treinamento: 10 a 100 árvores.

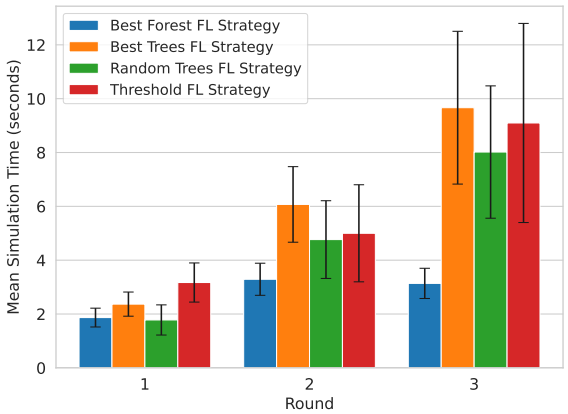

Figura 5. Tempo de treinamento: 110 a 200 árvores.

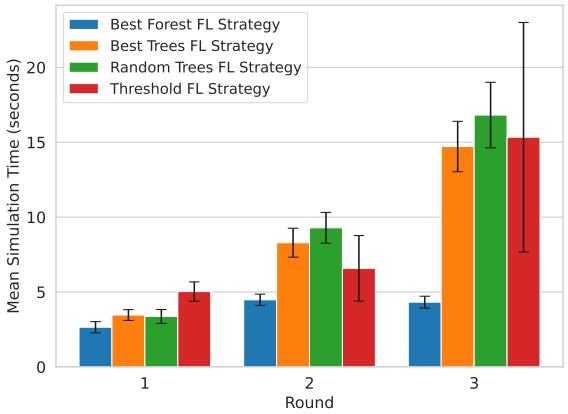

Figura 6. Tempo de treinamento: 210 a 300 árvores.

6. Conclusões e Trabalhos Futuros

A Internet das Coisas depende de nós de computação na borda para processar dados mais próximos de sua fonte, como sensores e atuadores. Esses nós oferecem mais poder computacional do que dispositivos IoT regulares e suportam o treinamento e a execução de modelos de aprendizado de máquina leves para aplicações IoT. O Aprendizado Federado emerge nesse cenário para oferecer uma camada adicional de proteção à privacidade dos dados.

Este artigo propõe uma solução, baseada nos princípios do Aprendizado Federado, chamada FEDT (Árvore de Decisão Federada), que aproveita o conhecimento combinado de múltiplos nós de borda para alcançar alta precisão sem centralizar dados.

Como trabalhos futuros, o FEDT pode ser aprimorado com a aplicação de algoritmos de poda, que são técnicas comuns usadas para reduzir a complexidade dos modelos de árvores de decisão, removendo ramos desnecessários e melhorando a generalização do modelo. A incorporação de algoritmos de poda ao FEDT pode resultar em modelos mais compactos e eficientes, o que é crucial para dispositivos IoT com recursos limitados de computação e armazenamento.

Além disso, um critério de seleção que considera conjuntos de dados não identificados pode ser desenvolvido para melhorar ainda mais a privacidade dos dados no contexto do aprendizado federado. Esse critério de seleção pode levar em conta informações estatísticas sobre os conjuntos de dados distribuídos entre os nós de borda, permitindo que o FEDT treine modelos de forma mais eficaz enquanto preserva a privacidade dos dados individuais. Essas melhorias potenciais podem tornar o FEDT uma solução ainda mais robusta e eficiente para aplicações de aprendizado federado em ambientes IoT.

Referências

Berman, J. J. (2016). Chapter 4 - understanding your data. In Berman, J. J., editor, *Data Simplification*, pages 135–187. Morgan Kaufmann, Boston.

Candanedo, L. M., Feldheim, V., and Deramaix, D. (2017). Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140:81–97.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520.

Hauschild, A.-C., Lemanczyk, M., Matschinske, J., Frisch, T., Zolotareva, O., Holzinger, A., Baumbach, J., and Heider, D. (2022). Federated Random Forests can improve local performance of predictive models for various healthcare applications. *Bioinformatics*, 38(8):2278–2286.

Huang, P., Li, D., and Yan, Z. (2023). Wireless federated learning with asynchronous and quantized updates. *IEEE Communications Letters*, 27(9):2393–2397.

Ji, Y. and Chen, L. (2023). Fedqnn: A computation–communication-efficient federated learning framework for iot with low-bitwidth neural network quantization. *IEEE Internet of Things Journal*, 10(3):2494–2507.

Lindskog, W. and Prehofer, C. (2023). A federated learning benchmark on tabular data: Comparing tree-based models and neural networks. In *2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 239–246.

Markovic, T., Leon, M., Buffoni, D., and Punnekkat, S. (2022). Random forest based on federated learning for intrusion detection. In Maglogiannis, I., Iliadis, L., Macintyre, J., and Cortez, P., editors, *Artificial Intelligence Applications and Innovations*, pages 132–144, Cham. Springer International Publishing.

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., and Foozy, C. F. M. (2021). Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE access*, 9:22351–22370.

Nettleton, D. (2014). Chapter 6 - selection of variables and factor derivation. In Nettleton, D., editor, *Commercial Data Mining*, pages 79–104. Morgan Kaufmann, Boston.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Shen, T., Mishra, C. S., Sampson, J., Kandemir, M. T., and Narayanan, V. (2022). An efficient edge-cloud partitioning of random forests for distributed sensor networks. *IEEE Embedded Systems Letters*.

Singhal, P., Pandey, S. R., and Popovski, P. (2024). Greedy shapley client selection for communication-efficient federated learning. *IEEE Networking Letters*, page 1–1.

Yu, R. and Li, P. (2021). Toward resource-efficient federated learning in mobile edge computing. *IEEE Network*, 35(1):148–155.