UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Pedro Lopes Maia

Prototipação de um Sistema de Localização utilizando Redes LoRaWAN

Recife

Março de 2024

Pedro Lopes Maia

Prototipação de um Sistema de Localização utilizando Redes LoRaWAN

Artigo apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE
Departamento de Estatística e Informática
Curso de Bacharelado em Sistemas de Informação

Orientador: Victor Wanderley Costa de Medeiros

Recife
Março de 2024

Prototipação de um Sistema de Localização utilizando Redes LoRaWAN

[Pedro Lopes Maia] [1], **[Victor Wanderley Costa de Medeiros]** [1]

[1]Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros, s/n, - CEP: 52171-900 – Recife – PE – Brasil

pedrolopes_maia@hotmail.com, victor.wanderley@ufrpe.br

Resumo. *Com a proliferação do uso de tecnologias IoT, soluções eficientes em termos de uso de bateria e aplicabilidade para posicionamento de dispositivos se tornaram cada vez mais necessárias devido à demanda por serviços baseados em localização. Nesse contexto, técnicas de localização baseadas em sinal, como o fingerprinting, representam uma solução muito apropriada por atenderem aos requisitos dessas aplicações. Neste estudo, empregou-se um conjunto de dados públicos contendo valores de RSSI provenientes de uma rede LoRaWAN para criar modelos de aprendizado de máquina com o intuito de avaliar sua eficácia no posicionamento de dispositivos LoRa, oferecendo uma alternativa ao GPS, que devido ao alto consumo de energia das baterias dos dispositivos, em muitas situações, não é viável para sistemas IoT. Após a devida avaliação de hiperparâmetros e aplicação de metodologias apropriadas para cada algoritmo estudado, foi obtido um modelo capaz de realizar previsões com erro médio de* $301,34$ *metros e mediana de* 164.26 *metros.*

Abstract. *With the proliferation of the use of IoT technologies, efficient solutions in terms of battery usage and applicability for device positioning have become increasingly necessary due to the demand for location-based services. In this context, signal-based localization techniques, such as fingerprinting, represent a very appropriate solution as they meet the requirements of these applications. In this study, a public dataset containing RSSI values from a LoRaWAN network was used to create machine learning models to evaluate their effectiveness in positioning LoRa devices, offering an alternative to GPS, which due to the high power consumption of device batteries, in many cases, is not viable for IoT systems. After evaluating hyperparameters and applying appropriate methodologies for each algorithm studied, a model was obtained capable of making predictions with an average error of* 301.34 *meters and a median of* 164.26 *meters.*

1. Introdução

Localização de ativos dispostos em áreas externas de grande extensão é uma demanda comum de algumas indústrias. Nesses casos, abordagens populares de posicionamento usadas no cotidiano de usuários domésticos, como o GPS ou tecnologias baseadas em celular, não podem ser empregadas devido ao alto consumo de energia nas baterias dos dispositivos, baixo alcance de sinal e alto custo de implementação. É por essa razão que sistemas de localização necessitam fazer uso de redes LPWAN. Redes LPWAN (Low

Power Wide Area Network) são ideais para aplicações IoT por atender aos seus principais requisitos: longo alcance de sinal, que pode ser entre 2 e 5 km [Centenaro et al. 2016], baixo consumo de energia, já que as baterias podem durar até dez anos com uma única carga [Patel and Won 2017], escalabilidade e manutenibilidade. Entre as redes LPWAN mais populares no mercado de IoT, a que tem relevância para o presente trabalho é a LoRaWAN, uma rede composta por dispositivos LoRa e organizada de acordo com um protocolo de mesmo nome.

Conforme demonstrado por [Anjum et al. 2022], o emprego de tecnologia LoRa junto à algoritmos de aprendizado de máquina no sentido de desenvolver soluções baseadas em RSSI e, portanto, independentes de GPS, para localização de ativos é uma abordagem viável. Além de adequada para a localização de dispositivos que compõe sistemas IoT, por esse tipo de sistema não poder depender do sistema de posicionamento global, por ser uma alternativa cara do ponto de vista energético. A técnica de posicionamento avaliada pelos autores, conhecida pela comunidade de pesquisa por *fingerprinting*, trata-se da modelagem de funções com o uso de algoritmos de aprendizado de máquina para mapear propriedades do sinal de uma rede, como o já citado RSSI (Received Signal Strength Indicator), e as coordenadas de dispositivos conectados a ela. O *fingerprinting* representa, então, uma solução para sistemas IoT de localização de ativos conectados via rede LoRaWAN que vale a pena ser pesquisada.

O presente trabalho tem como objetivo a avaliação de algoritmos de aprendizado de máquina para localização de dispositivos LoRa. Foi utilizado um conjunto de dados público de mensagens LoRa coletados em uma extensa área urbana, que foram ordenadas de acordo com as datas de envio, e utilizadas no desenvolvimento de modelos com os algoritmos k-nearest neighbors, Random Forest e rede neural RBF. Os algoritmos foram treinados com diferentes partições dos dados da base original, criadas para representar mensagens acumuladas a partir de uma data inicial. Seus desempenhos foram avaliados com cada partição de modo a compreendermos a evolução das previsões de coordenadas ao longo do tempo, conforme os algoritmos são treinados com novos dados.

O restante do artigo está dividido como segue. A seção 2 introduz as redes LPWAN e LoRaWAN, além dos algoritmos utilizados. A seção 3 apresenta resumidamente os trabalhos relacionados. A seção 4 descreve os métodos utilizados no trabalho, ferramentas, dados e materiais, apresentando o conjunto de dados que foi empregado, além da metodologia aplicada no desenvolvimento dos modelos de aprendizado de máquina. A seção 5 discute em detalhes os resultados que foram obtidos e, finalmente, a seção 6 resume este trabalho com uma breve discussão das conclusões tiradas.

2. Referencial teórico

2.1. Fingerprinting baseado em aprendizado de máquina

Aprendizado de Máquina [Géron 2019c] é um campo da ciência da computação e da inteligência artificial em que são estudados conjunto de regras e procedimentos que proporcionam aos computadores a capacidade de agir e tomar decisões com base em dados, ao contrário de serem diretamente programados. Com métodos estatísticos empregados por algoritmos de aprendizado de máquina, treinados com dados que representam características de entidades do mundo real, obtêm-se modelos preditivos que auxiliam na

tomada de decisão de organizações e que viabilizam o desenvolvimento de sistemas inteligentes capazes de adaptar-se ao comportamento de usuários ou mudanças do ambiente onde atuam. O aprendizado de máquina tem evoluído bastante e é resultado de muitos avanços tecnológicos nos últimos anos, podendo ser visualizado em alguns projetos inovadores como carros que dirigem, visão computacional e sistemas de reconhecimento de voz.

No campo de estudo de localização interna e navegação, *fingerprinting* é o nome de uma técnica de posicionamento de pessoas ou objetos. Sua metodologia compreende a coleta de dados de dispositivos conectados, seguido pelo seu processamento e armazenamento, criando um mapeamento entre os metadados da rede e a posição dos dispositivos no banco de dados. A localização de dispositivos é uma estimativa realizada através da comparação entre as propriedades da rede ao qual estão conectados, e os registros antigos. Modelos preditivos podem ser então desenvolvidos para realizarem previsões de localizações de dispositivos, após os treinamentos realizados com os dados coletados. Esse procedimento é conhecido como *machine learning-based fingerprinting*

2.2. Low Power Wide Area Networks

Low Power Wide Area Network (LPWAN) é um tipo de rede de comunicação sem fio cujas principais características são o longo alcance e baixo consumo de energia. Ideal para dispositivos conectados que precisam enviar pacotes pequenos, durante intervalos de tempo irregular e por grandes distâncias. A taxa de dados de uma rede do tipo LPWAN é menor que 5000 bits por segundo [LAB'S 2016]. O gráfico da Figura 1 mostra uma comparação entre redes LPWAN e outros tipos de rede no tocante a alcance e largura de banda.

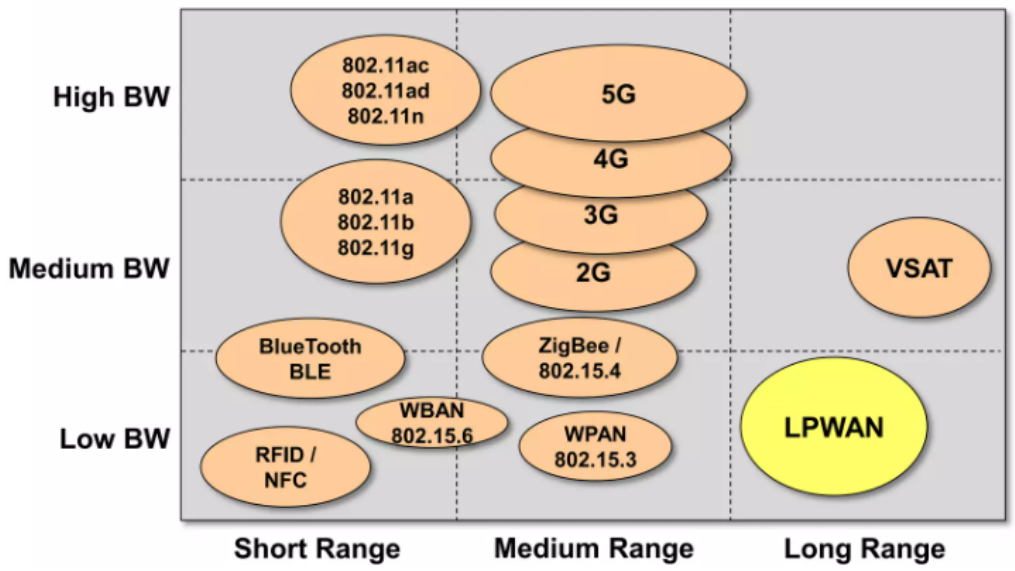


Figura 1. Gráfico comparativo de redes sem fio FONTE: PETER R. EGLI, 2023, http://www.slideshare.net/PeterREgli/lpwan

A baixa taxa de dados permite que o dispositivo consiga dispor de maior quantidade de energia ao enviar um bit dentro de um pacote, o que aumentam as chances da informação chegar ao seu destino, sobrevivendo à obstáculos que possa encontrar pelo caminho e assim alcançando grandes distâncias. Essa relação entre o alcance e taxa de dados

é explicada pelo teorema de Shannon-Hartley, que determina a quantidade de informação que pode atravessar um canal de comunicação com determinada largura de banda.

Os dispositivos conectados a redes LPWAN comumente utilizam duas topologias: estrela e malha. A topologia estrela é uma arquitetura que compreende um conjunto de dispositivos posicionados dentro da área de cobertura e uma torre de comunicação no seu centro, responsável por repassar o sinal capturado dos dispositivos para o servidor da rede. É geralmente preferida em relação à malha devido à preservação da energia da bateria e à expansão da faixa de comunicação. A topologia malha é composta por estações, nós e roteadores, todos conectados uns aos outros, com nós trocando mensagens entre eles e entre as estações. Essa composição pode levar a grande complexidade, aumento de latência e custos, com a redundância comprometendo a eficiência energética da rede. Porém, para algumas aplicações, essa topologia pode ser preferível por sua fácil escalabilidade e maior alcance devido ao uso de roteadores.

Redes do tipo LPWAN são projetadas para serem flexíveis e escaláveis, o que permite que possam atender a diversos casos de uso cujos requisitos podem variar entre capacidade, cobertura, custo, consumo de energia ou algo específico ao seu domínio. Cidades inteligentes, medição inteligente, automação industrial e varejo são exemplos de projetos de IoT que possuem como prioridade implementação de baixo custo, enquanto que agricultura inteligente e sistemas de segurança demandam por menor consumo de energia [Chaudhari and Zennaro 2020].

2.3. Redes LoRaWAN

LoRaWAN é o nome do protocolo de comunicação de redes LPWAN que são baseadas em dispositivos LoRa, assim sendo capazes de atender a todos os requisitos de aplicações IoT mencionados na seção anterior. As especificações das redes LoRaWAN são mantidas pela *Lora Alliance*, que se define como "uma associação aberta e sem fins lucrativos que se tornou uma das maiores alianças e de mais rápido crescimento no setor de tecnologia desde a sua criação em 2015"[Alliance 2023]. A arquitetura LoRaWAN tem evoluído desde sua criação para melhor acompanhar as necessidades do mercado. Inicialmente contando apenas com as camadas físicas, de comunicação e de aplicação na sua primeira versão, a *Lora Alliance* incluiu a especificação de interface de backend no ano de 2017, para definir o fluxo de mensagens entre as entidades da rede [Lora Alliance].

As Redes LoRaWAN são implementadas em topologia estrela [Yegin et al. 2020], um modelo que demanda um ponto de acesso (uma estação, ou "gateway") designado para abranger uma área e receber mensagens de dispositivos dispostos nela. A Figura 2 ilustra a arquitetura de uma rede LoRaWAN. A camada física representa o conjunto de dispositivos utilizados, e a camada de conexão os protocolos de comunicação entre os dispositivos e as estações. A comunicação entre os dispositivos e as estações ocorrem de forma bidirecional, sendo o envio de *payloads* dos dispositivos a mais frequente. As estações estabelecem conexões TCP/IP com os servidores centrais da rede, que por sua vez reenviam os pacotes recebidos para o servidor da aplicação que fará uso deles.

Os pacotes do tráfego entre os servidores da rede e da aplicação são protegidos por criptografia simétrica. Os dispositivos possuem chaves específicas que são utilizadas pelo servidor durante a requisição para o estabelecimento da conexão com o servidor da aplicação. Quando a requisição é aceita, o servidor de junção, ou *join server*, cria uma

chave de sessão derivada da chave do dispositivo e as armazena durante o processo.

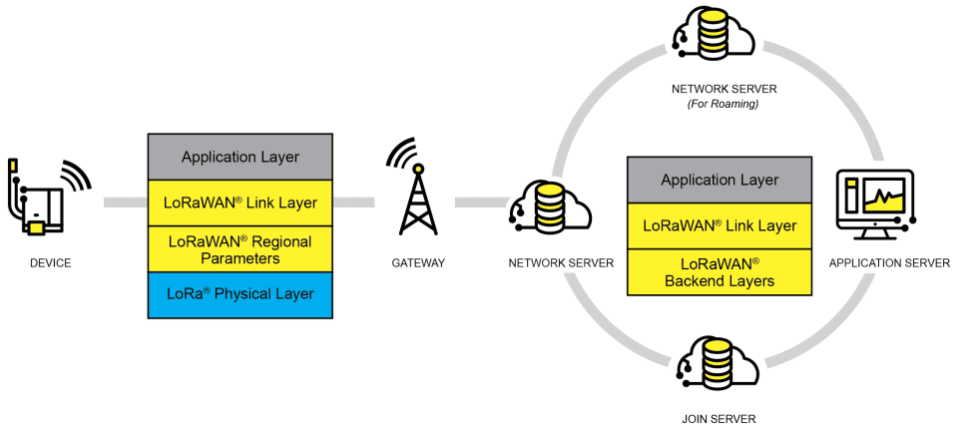


Figura 2. Arquitetura de uma rede LoRaWAN. FONTE: Lora Alliance, 2023, https://lora-alliance.org/about-lorawan.

2.4. K-Nearest Neighbors

O K-Nearest Neighbors (KNN) é um algoritmo de aprendizado de máquina supervisionado, cujos hiperparâmetros são um número natural, e maior que zero, *k* e uma métrica de distância, como distância euclidiana, distância de Manhattan ou distância de Minkowski. Seja $\overline{X}$ um conjunto de treinamento, cujos pontos $\overline{X_1}, \ldots, \overline{X_n}$ são vetores pertecentes a um espaço multidimensional de atributos. O algoritmo classifica o ponto de entrada calculando todas as distâncias entre ele e os demais pontos do espaço de atributos, e determinando seu rótulo a partir dos rótulos da maioria dos *k* pontos mais próximos, conforme descrito no seguinte pseudocódigo:

1. Carregar os dados de treino e teste.
2. Escolha do valor de k
3. Para cada ponto nos dados de teste:
   (a) Achar todas as distâncias para cada ponto de dados
   (b) Armazenar as distâncias em uma lista e ordená-la
   (c) Determinar uma classe para a amostra em função da classe da maioria dos k pontos mais próximos

Em problemas de regressão, o algoritmo calcula a média dos valores dos k pontos mais próximos dentro do espaço multidimensional para determinar o valor da instância de entrada:

$$f_{KNN}(x') = \frac{1}{k} \sum_{i \in N_k(x')} y_i$$

Apesar de sua fácil implementação e rápido processamento durante o treinamento, devido ao fato de que o computador não precisa calcular nenhum valor ou parâmentro durante esse processo, o KNN possui como principal desvantagem a falta de interpretabilidade, sendo assim impossível saber o grau de influência dos atributos sobre a previsão.

2.5. Árvore de decisão

Versátil e de fácil compreensão, a árvore de decisão [Géron 2019a] é um método de aprendizado de máquina não-paramétrico que pode ser aplicado em problemas de classificação e regressão. Trata-se de uma estrutura composta por nós internos e folhas dispostos em vários níveis de profundidade. O nó do nível superior e os do último nível são chamados de raiz e folhas, espectivamente. As folhas não possuem nós filhos, e todos os demais nós possuem no mínimo uma folha. Cada nó representa um subconjunto dos dados de treino que obedece à uma determinada condição associada a um atributo dos dados. Invariavelmente, os nós possuem como propriedades a quantidade de instâncias do subconjunto representado e a condição que obdecem.

Em modelos de classificação, os nós representam uma classe e possuem um valor de gini, uma métrica do grau de pureza daquele subconjunto. Um subconjunto é considerado puro, e portanto, tem valor de gini igual a zero, se todas as suas instâncias tiverem a mesma classe. A métrica Gini, exibida na equação 1, quantifica a chance de um teste selecionado aleatoriamente ser classificado de forma errada por um algoritmo de árvore de decisão, variando de 0 (absolutamente puro) a 1 (totalmente impuro). Trata-se de uma medida da impureza ou entropia nos valores de um conjunto de dados. Em modelos de regressão, os nós possuem um valor de *MSE*, "Mean Squared Error", uma métrica usada para avaliação de modelos desse tipo, e a média dos valores das instâncias do subconjunto.

$$GINI = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{1}$$

Para o desenvolvimento do modelo, foi aplicada a implementação da árvore de decisão disponível na biblioteca scikit-learn, que se baseia no algoritmo CART (Classification and Regression Tree) para fazer seu treinamento. Seja k um atributo do conjunto de dados e t_k um limiar para esse atributo, algoritmo CART funciona da seguinte forma:

1. Para cada valor de k, encontrar o t_k que resulte no menor valor na equação 2, caso se trate de um problema de regressão, ou na equação 3 para problemas de classificação.
2. Com base nos pares encontrados na etapa anterior, dividir o conjunto de dados de acordo com o par (k, t_k) que origine subconjuntos com maior grau de pureza.
3. Repetir a etapa 1 ate que algum critério de parada seja satisfeito.

Selecionando de forma aleatória um atributo k e um valor limiar t_k, o algoritmo procura os pares (k, t_k) que separam o conjunto de dados em duas partes de acordo com a comparação com esses pares, de forma a minimizar a função de custo (1), para que os dois subconjuntos tenham o menor MSE possível. m representa a quantidade total de instâncias no conjunto de dados, e m_{esq} e m_{dir} as quantidades de instâncias nos subconjuntos dos nós à esquerda e à direita do nó pai, respectivamente. Após a primeira separação dos dados, o algoritmo recursivamente executa divisões nos subconjuntos resultantes, até que a árvore alcance a profundidade máxima estabelecida.

$$J(k, t_k) = \frac{m_{esq}}{m} MSE_{esq} + \frac{m_{dir}}{m} MSE_{dir} \tag{2}$$

$$J(k, t_k) = \frac{m_{esq}}{m} Gini_{esq} + \frac{m_{dir}}{m} Gini_{dir} \quad (3)$$

As previsões de um modelo de regressão são feitas através da comparação entre os atributos e valores das instâncias de entrada com os pares (k, t_k) de cada nó, desde a raiz até as folhas. A depender do resultado da comparação, o algoritmo executará uma nova comparação com o nó filho da esquerda ou da direita. As folhas possuem as mesmas propriedades dos nós, menos a condição. Ao alcançar a folha, o algoritmo devolve a média dos valores de seu subconjunto como resultado da previsão.

O processo de construção de uma árvore de decisão treinada para classificação funciona de forma similar. Porém, nesse caso o algoritmo procura minimizar a função de custo (2). O objetivo é criar dois subconjuntos com menor impureza possível. A previsão também é determinada pelo percurso do algoritmo entre a raiz e a folha, sendo a classe da instância determinada pela última. Ou seja, cada folha representa uma classe do problema, ou um valor real, em caso de regressão.

2.6. Random Forest

A fim de obter melhor performance, é comum o uso do método de aprendizado em conjunto, ou "ensemble learning", que consiste no treinamento de múltiplos modelos de forma agregada, com o uso de um ou vários algoritmos de aprendizado, para obter um modelo mais poderoso a partir da combinação de suas previsões, como é o caso do Random Forest, que funciona da seguinte maneira:

1. Contrua uma coleção de árvores com subconjuntos aleatórios de dados.
2. Realizar previsões com cada uma das árvores.
3. Retornar a previsão da maioria das previsões das árvores, ou a média dos resultados das previsões em caso de regressão.

O algoritmo Random Forest [Géron 2019b] é uma composição de árvores de decisão, que aplica o método bagging no processo de treinamento. Para cada uma das árvores, o algoritmo seleciona um subconjunto de *m* intâncias de treinamento, em que *m* é o número total de instâncias no conjunto de treinamento. As instâncias são escolhidas de forma aleatória e com repetição. O grupo de instâncias não utilizadas no treinamento da árvore é conhecido como "out-of-bag"e são aplicadas na avalização do modelo. A performance do algoritmo é obtida a partir do cálculo da média da performance de cada uma das árvores. As previsões também representam a média das previsões das árvores.

As árvores de decisão que compõe o random forest realizam as separações dos dados escolhendo o melhor atributo entre um subconjunto aleatório de atributos, o que aumenta a diversidade entre as árvores e possibilita a medição do grau de importância dos atributos. O cálculo da importância dos atributos é realizado medindo-se o quanto que os nós que utilizaram o atributo para separação dos dados reduz a impureza, ou MSE, dos subconjuntos que separou.

2.7. Redes neurais RBF

Uma Rede Neural RBF [Aggarwal 2018] é uma rede neural que emprega funções RBF como ativações nas suas camadas ocultas, caracterizando-se por essa característica. Essa

rede neural é comumente organizada em três níveis: a camada de entrada, a camada oculta, onde as funções RBF são aplicadas, e a camada de saída. A figura 3 ilustra a arquitetura de uma rede RBF.

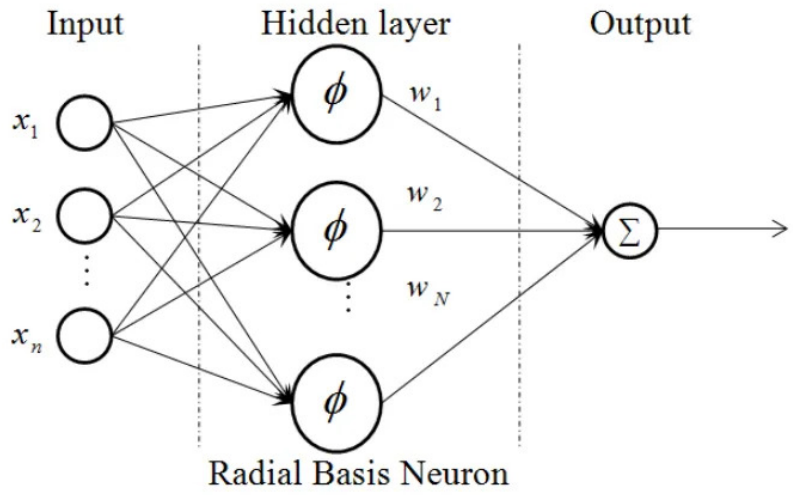


Figura 3. Arquitetura de uma rede RBF. FONTE: Medium, 2023, https://medium.com/@parakatta/radial-basis-function-e3121f85a29a.

Cada uma das *m* unidades de ativação da camada oculta contém um protótipo de vetor μ de dimensões iguais as dos pontos de treino, e que são aprendidos de maneira não-supervisionada. Uma abordagem comum, e que foi utilizada no presente trabalho, é a aplicação do algoritmo K-Means para particionar os dados em *m* clusters e empregar seus centroids como protótipos de vetores. Dessa forma, o número de unidades de ativação na camada oculta determina como os dados serão clusterizados. Outro hirperâmetro importante é a largura de banda σ, uma variável que determina o grau de influência da unidade de ativação sobre os pontos dos clusters mais próximos de seu protótipo de vetor. A função de ativação hi é definida de acordo com a equação a baixo:

$$h_i = \phi_i(\overline{X}) = exp\left(- \frac{\left|\left|\overline{X} - \mu_i\right|\right|^2}{2\sigma_i^2} \right) \quad \forall_i \in \{1, ..., m\} \tag{4}$$

$$Erro = \frac{1}{2} \left|\left| H\overline{W} - \overline{y} \right|\right|^2 \tag{5}$$

A última camada existe apenas para treinar o vetor $\overline{W}$, que contém os pesos das conexões com a camada oculta, e computar a saída. Supondo-se que o conjunto de treino $\overline{X}$ seja composto por n pontos de treino, $\overline{X_1}$, ..., $\overline{X_n}$, após processados pela camada oculta, os pontos de treino X_i se transformam em vetores de dimensão m, H1, ..., Hn, que ao serem empilhados, formam a matriz H de proporções nxm . Além disso, os valores observados são expressos na forma de vetor y = $[y_1, \ldots, y_n]^T$. A previsão da rede, que chamaremos de $\overline{y}$, é feita pela operação $\overline{W}$H. Dessa forma, o aprendizado da última camada ocorre através da otimização da função de custo (5), cujos resultados calculados em cada iteração são utilizados para reajustar os pesos da rede.

3. Trabalhos Relacionados

Apesar de o método *fingerprinting* ser mais frequentemente estudado para localização em ambientes internos, a avaliação de algoritmos de *machine learning* para determinar localizações em áreas externas tem ganhado alguma popularidade ao longo dos últimos anos. Com o propósito de contribuir nesse sentido, [Aernouts et al. 2018] construiu três conjuntos de dados LPWAN, sendo dois com redes Sigfox e um em uma rede LoRaWAN. O conjunto de dados LoRaWAN consiste em mensagens coletadas a partir de dispositivos LoRa anexados em carros do serviço postal da cidade da Antuérpia, contendo metadados da comunicação entre os dispositivos LoRa e as 68 estações espalhadas pela área central da cidade, além das coordenadas auferidas por dispositivos GPS. As mensagens contêm dados sobre os RSSIs da comunicação com as estações, o fator de espalhamento, o HDOP, o horário do envio da mensagem e as coordenadas do dispositivo. Durante um período de quase três meses, de novembro de 2017 até fevereiro de 2018, a base de dados agregou 123529 pacotes. A usabilidade desse conjunto de dados foi avaliada com o algoritmo KNN, em que se descobriu que, com k = 11, o algoritmo pode identificar a localização de um dispositivo LoRa com um erro médio de 398,4 metros e mediana de 273,03 metros.

Dando continuidade ao trabalho anterior, [Janssen et al. 2020] realizaram avaliações de algoritmos de aprendizado de máquina apropriados para problemas de regressão numa versão atualizada do conjunto de dados apresentado anteriormente. Os dados de entrada foram transformados de acordo com quatro formas de pré -processamento: positiva, normalizada, exponencial e potencial. Também foi aplicado PCA no conjunto de dados, para diminuir a quantidade de dimensões e o tempo de execução do treinamento. Entre todos os algoritmos testados, o KNN ponderado e o Random Forest foram os que apresentaram os melhores resultados para os dados representados na forma potencial, com erro médio de 343 metros e 340 metros, respectivamente.

[Anagnostopoulos and Kalousis 2019] também apresentaram novos resultados relativos à avaliação de *fingerprinting* baseado em *machine learning* no conjunto de dados coletados na Antuérpia, sendo os primeiros a demonstrarem a eficiência da aplicação de *deep learning*. Empregando a mesma divisão de dados em subconjuntos de treino/validação/teste que os autores do trabalho citado anteriormente, eles alcançaram um desempenho aproximado com o algoritmo KNN, chegando a um erro médio de 391 metros, calculados segundo a distância euclidiana, para os dados de cada tipo de pré-processamento. Com uma rede neural, os autores obtiveram um erro médio de 357 metros e mediana de 206 metros no subconjunto de teste.

4. Materiais e métodos

4.1. Conjunto de dados

O conjunto de dados utilizado no presente trabalho foi construído por [Aernouts et al. 2018], e consiste em uma coleção de mensagens de dispositivos LoRa, coletadas durante um período de três meses na região central da cidade de Antuérpia, na Bélgica. Dispostivos foram acoplados em carros dos correios pelos pesquisadores, para que suas coordenadas pudessem ser enviadas ao servidor, com metadados da rede LoRaWAN, pela qual os dispositivos estavam conectados às estações expalhadas pelo perímetro urbano. São ao todo 127038 mensagens com os valores de RSSI (Received Signal Strength Indicator) de cada uma das 44 estações. O RSSI é um

número inteiro que serve como indicativo da força do sinal do link da comunicação, que pode variar conforme a distância ou devido à presença de obstáculos no caminho das mensagens, tais como prédios ou montanhas. Para representar uma comunicação mal sucedida entre o dispositivo e uma estação, convencionou-se utilizar o valor de RSSI -200 para esses casos. A relação completa dos atributos presentes no conjunto de dados é exibida na tabela 1.

Atributo	Tipo de dado
RSSI	Número inteiro
Diluição Horizontal de Precisão	Número real
Endereço do dispositivo LoRaWAN	Texto
Endereço do dispositivo EUI	Texto
Fator de espalhamento	Número real
Canal	Número inteiro
Tempo no ar	Número real
Payload	Texto
ADR	Número inteiro
Contador	Número inteiro
Latitude	Número real
Longitude	Número real
ESP	Número real
SNR	Número real
Tipo de timestamp	Texto
Tempo	Texto

Tabela 1. Atributos do conjunto de dados

Entre as versões do material disponibilizadas pelos autores, utilizamos a mais recente, que compreende o conjunto de dados em formatos CSV e JSON, e um arquivo com a localização das estações. A versão do conjunto de dados em JSON possui os mesmos metadados da versão em CSV, mais atributos dos dispositivos presentes na rede. Entre todos os atributos presentes no arquivo JSON, aqueles que foram usados no experimento foram os RSSI, fatores de espalhamento, datas dos envios das mensagens, identificadores dos dispositivos e suas coordenadas.

O fator de espalhamento é uma dos parâmetros configuráveis do padrão LoRa. O aumento do fator de espalhamento em uma comunicação LoRa resulta em uma comunicação mais robusta e com maior alcance, mas ao custo de uma taxa de transmissão mais baixa, maior tempo de transmissão e maior consumo de energia. A escolha do fator ideal deve ser feita com base nos requisitos específicos da aplicação em questão.

A fim de otimizar a qualidade da comunicação e o consumo de energia, as redes LoRaWAN adaptam dinamicamente o fator de espalhamento dos seus dispositivos por meio de um algoritmo que mede o ruído de sinal, o ADR (Adaptive Data Rate). Na Figura 4 é exibido um gráfico de torta representando a proporção de mensagens por fator de espalhamento no conjunto de dados utilizado neste trabalho. No total há 6 seis fatores de espalhamento possíveis do SF7 ao SF12. O fato de que as mensagens do conjunto de

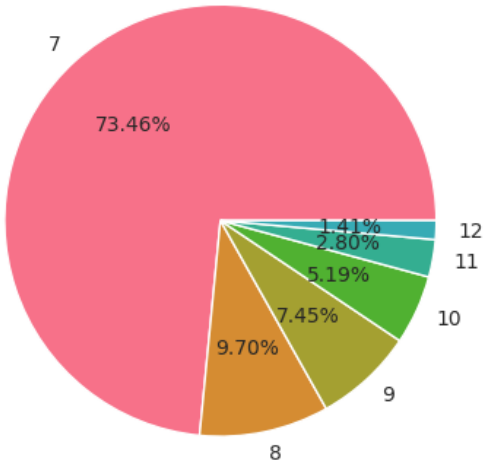


Figura 4. Fatores de espalhamento

dados foram coletadas em uma área de apenas 204,51 km^2 justifica a grande proporção de mensagens com fator de espalhamento igual a 7 (menor alcance, menor consumo e maior taxa de dados).

Após análise dos metadados presentes no arquivo JSON, ficou evidente uma desproporcionalidade entre as quantidades de mensagens enviadas por cada dispositivo, e as quantidades de mensagens enviadas por dia durante os meses de preparação do conjunto de dados. Na Figura 5, o gráfico mostra que houve um aumento expressivo no número de mensagens enviadas pelos dispositivos a partir do ano de 2019, contrastando com a pouca quantidade de mensagens enviadas no ano anterior. A contribuição de cada dispositivo para o envio de mensagens é apresentada na Figura 6.

4.2. Pré-processamento

A representação dos dados escolhidos para este trabalho foi apresentada por [Torres-Sospedra et al. 2015] em um estudo sobre *fingerprinting* em dados de mensagens de redes WiFi. A representação normalizada é obtida após uma transformação linear que mapea os dados para um limite entre [0,1], a partir do escalonamento dos dados na forma positiva, como mostrado na Equação 6.

$$Normalized_i(x) = \frac{Positive_i(x)}{(-\tau)} \tag{6}$$

A forma positiva é obtida após subtrair todas as entradas por um limiar τ, como mostrado na Equação 7. [Torres-Sospedra et al. 2015] sugerem que τ seja igual ao menor valor de RSSI, para que a influência de sinais que não foram recebidos seja anulada, já que todo valor igual ou menor que τ é convertido em 0. Assim, no presente estudo, a transformação positiva foi feita com $\tau = -200$.

$$Positive_i(x) = x - \tau \tag{7}$$

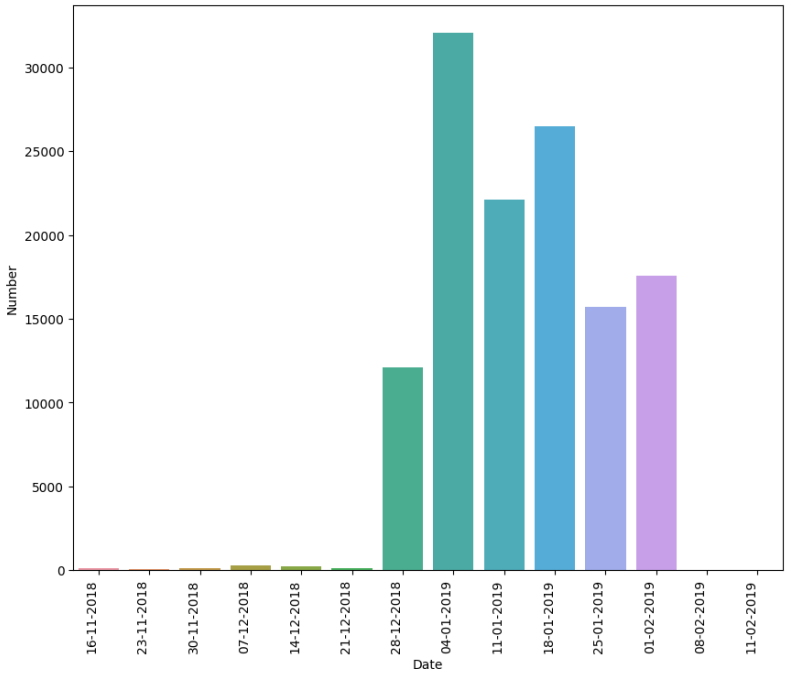


Figura 5. Quantidade de mensagens por semana

Por fim, para diminuir o ruído, complexidade e tempo de treinamento foi aplicada um método chamado "Principal Component analysis" (PCA) sobre os dados. No campo de aprendizado de máquina e estatística, o conceito de dimensionalidade é utilizado para descrever a quantidade de características ou atributos presentes em um conjunto de dados. O PCA é uma técnica de redução de dimensionalidade linear, em que os dados são projetados para um espaço dimensional menor, preservando as informações essenciais contidas no conjunto original. A aplicação de PCA se faz necessária principalmente devido à esparsidade dos dados, pois, poucas estações podem contribuir considerávelmente para o aprendizado dos algoritmos. Após a aplicação do PCA, a dimensionalidade dos dados foi reduzida de 45 para 40. A escolha de 40 atributos foi motivada pelo trabalho de [Janssen et al. 2020] realizado com uma versão diferente do mesmo conjunto de dados.

4.3. Configuração do experimento

Todos os experimentos foram realizados com código escrito na linguagem Python, particularmente a versão 3.10.12. As bibliotecas gratuitas scikit-learn (1.2.2) [Pedregosa et al. 2011] e Keras (2.12.0) [Chollet et al. 2015] foram utilizadas para a criação dos modelos de aprendizado de máquina. O código da implementação do modelo da rede RBF foi disponibilizado por [Vidnerova 2019], e pode ser acessado pelo link disponibilizado nas referências.

Do conjunto de dados foram descartadas todas as mensagens de dispositivos que enviaram menos de 5000 delas, no que resultou na eliminação das mensagens enviadas em 2018. Tal medida foi aplicada por acreditarmos que poderia contribuir para obteção de melhores resultados dos algoritmos. Após o ordenamento das mensagens, de acordo com suas datas de envio, 13 subconjuntos de dados foram derivados, contendo múltiplos de 10000 mensagens ou a quantidade total. Os subconjuntos foram classificados de acordo

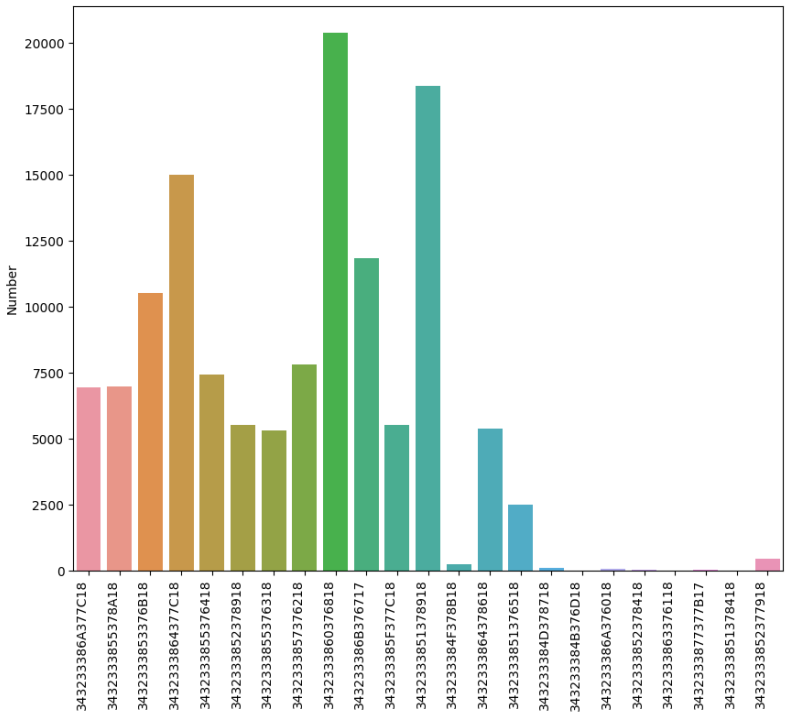


Figura 6. Quantidade de mensagens por dispositivo

com a data de envio das suas últimas mensagens, e utilizados para o treinamento dos algoritmos estudados na pesquisa, a fim de avaliar suas mudanças de performance ao longo dos períodos de tempo.

Os modelos desenvolvidos com rede neural RBF foram criados segundo uma prática comum em projetos de aprendizado de máquina, que é a separação dos dados em conjuntos de treino, validação e testes, em que o conjunto de validação é utilizado como conjunto de testes até serem encontrados valores adequados para os hiperparâmetros do algoritmo estudado, com o conjunto de testes sendo reservado para a avaliação final do modelo.

$$R^2 = 1 - \frac{\sum_{(y_i - \hat{y})^2}}{\sum_{(y_i - \bar{y})^2}} \tag{8}$$

Como o objetivo do trabalho é produzir modelos que possam estimar as coordenadas, latitude e longitude, de um dispositivo LoRa com a menor margem de erro possível, o erro médio e o erro mediano das previsões são as principais métricas utilizadas. O cálculo das distâncias é feito por uma função fornecida pela biblioteca gratuita geopy (2.3.0), que calcula a distância geodésica entre duas coordenadas, a distância mais curta na superfície de um modelo elipsoidal da Terra. O coeficiente de determinação, exibido na equação 8, também foi utilizado para avaliação do desempenho do modelo.

5. Resultados e discussões

5.1. K-Nearst Neighbors

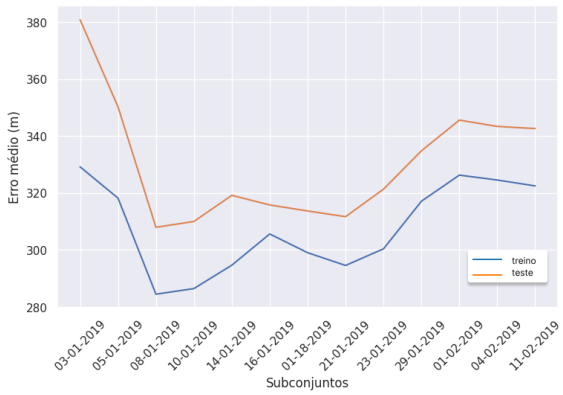
A quantidade de mensagens dos primeiros subconjuntos de dados utilizados para o treinamento tornou a prática comum de divisão dos dados em conjuntos de treino, validação e teste, pouco eficiente, devido a pouca quantidade de mensagens no conjunto de treino. Uma busca exaustiva com *grid-search* foi aplicada a fim de obter hiperparâmetros adequados para um bom desempenho. O desenvolvimento do modelo com *grid-search* consiste na aplicação do método *k-fold cross-validation* com uma lista de hiperparâmetros fornecidos. Primeiramente, os dados são separados em k subconjuntos de mesmo tamanho onde $k-1$ subconjuntos são utilizados para o treinamento do algoritmo, utilizando todas as combinações dos hiperparâmetros escolhidos. O subconjunto restante é utilizado como conjunto de teste, para avaliação. Esse processo é repetido k vezes, com todas os subconjuntos. Uma média do desempenho do modelo nas rodadas de treinamento é calculada para avaliar seu desempenho geral.

Durante os treinamentos com os subconjuntos, o KNN foi avaliado com a pontuação R^2 como métrica para escolher o melhor valor de K, por essa razão que os valores de R^2 apresentam uma tendência crescente no gráfico da Figura 7(b). Foram avaliados valores de K entre 1 e 60 e a fórmula de Minkowski foi utilizada para cálculo das distâncias. O modelo treinado com os dados enviados até o dia 08/01/2019 foi o que obteve melhor desempenho, como é possível observar na Tabela 2. Uma comparação entre os gráficos das Figuras 7(a) e 7(b) mostram que não existe uma relação direta entre os valores de R^2 e erro médio das previsões, pois, apesar do erro médio oscilar entre os dias 16/01 e 11/02, a pontuação R^2 se manteve crescendo a partir do dia 16/01. A Figura 7(c) apresenta a mediana dos erros.

Na figura 7(b), é possível observar que a partir da rodada de treinamento executada com as mensagens coletadas até o dia 16/01/2019, os valores de R^2 dos modelos aumentam proporcionalmente tanto com os dados de treino com os dados de testes. Os gráficos das figuras 7(a) e 7(c) demonstram que os modelos criados com KNN apresentaram suas melhores performances entre as rodadas de treinamentos dos dias 08/01/2019 e 21/01/2019.

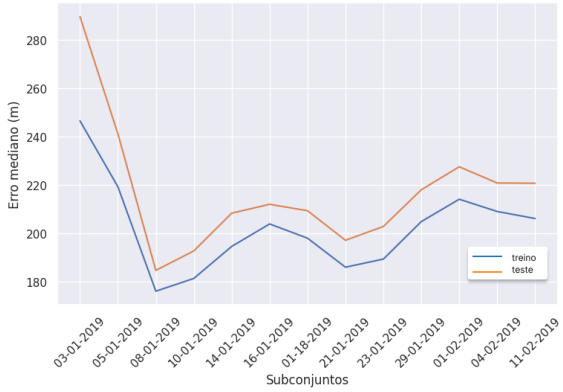
Intervalos	K	R^2	Erro médio	Erro mediano
03/01/2019	6	0.8867	380,83 m	289,62 m
05/01/2019	9	0.8855	350.26 m	241.19 m
08/01/2019	12	0.8907	307.97 m	184.79 m
10/01/2019	12	0.8875	310.01 m	192.81 m
14/01/2019	12	0.8904	319.19 m	208.42 m
16/01/2019	17	0.8950	315.82 m	212.10 m
18/01/2019	16	0.8965	313.70 m	209.44 m
21/01/2019	16	0.8992	311.70 m	197.22 m
23/01/2019	14	0.9011	321.35 m	202.93 m
29/01/2019	16	0.9073	334.81 m	218.12 m
01/02/2019	14	0.9099	345.60 m	227.61 m
04/02/2019	17	0.9112	343.39 m	220.90 m
11/02/2019	16	0.9129	342.63 m	220.80 m

Tabela 2. Resultados das rodadas de treinamento do modelo kNN



(a) Erros médios por subconjuntos - kNN

(b) R² por subconjuntos - kNN



(c) Erros medianos por subconjuntos - kNN

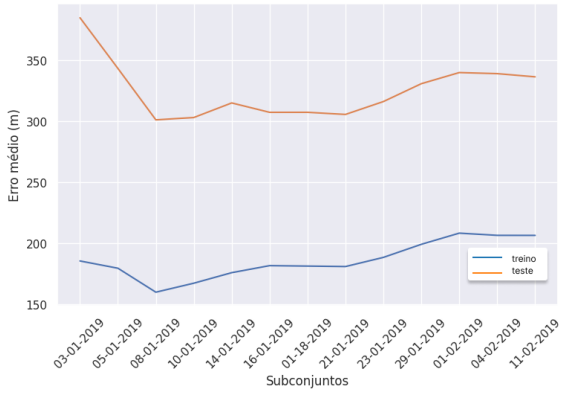
Figura 7. Resultados do modelo kNN

5.2. Random Forest

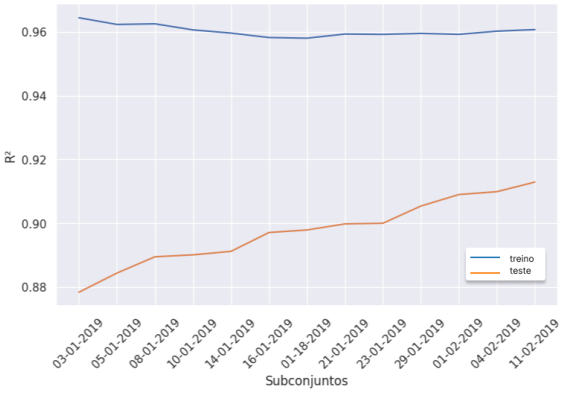
A técnica de validação cruzada, explicada na seção anterior, também foi utilizada para o desenvolvimento dos modelos com o algoritmo random forest. Durante este processo, foram avaliadas as seguintes quantidades de árvores: 25, 50, 75 e 100. Todos os modelos tiveram melhor resultado com 100 estimadores, apresentando desempenho similar aos modelos criados com KNN. O subconjunto que contém as mensagens coletadas até a data 08/01/2019 também proporcionou o modelo com os melhores resultados, com erro médio de $301,34$ metros com os dados de testes. Os gráficos das Figuras 8(a) e 8(c) exibem curvas muito afastadas em comparação com os resultados obtidos com os outros algoritmos, o que se explica pelo fato de que o random forest é mais propenso ao *overffiting*, quando o desempenho de um algoritmo com os dados de testes são consideravelmente piores do que com os dados de treino. Porém, os resultados exibidos na Tabela 3 demonstram que apesar disso, o random forest obteve melhor performance geral com os mesmos dados. O gráfico da Figura 8(b) exibe o crescimento da pontuação R^2 com os dados de testes ao longo das rodadas de treinamento, enquanto que os resultados R^2 com os dados de treino se mantiveram altos com certa estabilidade, próximo do valor $0,96$, em média.

Intervalos	R^2	Erro médio	Erro mediano
03/01/2019	0.8784	384.97 m	283.98 m
05/01/2019	0.8844	343.43 m	224.07 m
08/01/2019	0.8895	301.34 m	164.26 m
10/01/2019	0.8901	303.23 m	176.23 m
14/01/2019	0.8912	315.18 m	194.75 m
16/01/2019	0.8971	307.49 m	191.43 m
18/01/2019	0.8979	307.50 m	191.35 m
21/01/2019	0.8998	305.77 m	182.13 m
23/01/2019	0.9000	316.32 m	187.53 m
29/01/2019	0.9054	330.95 m	204.18 m
01/02/2019	0.9090	340.06 m	211.37 m
04/02/2019	0.9099	339.19 m	207.99 m
11/02/2019	0.9129	336.57 m	205.95 m

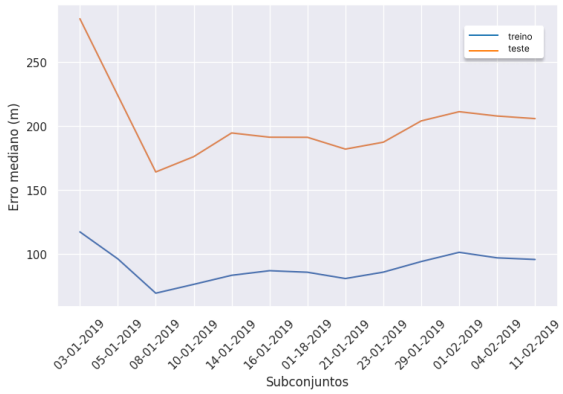
Tabela 3. Resultados das rodadas de treinamento do modelo Random Forest



(a) Erros médios por subconjuntos - Random Forest



(b) R^2 por subconjuntos - Random Forest



(c) Erros medianos por subconjuntos - Random Forest

Figura 8. Resultados do modelo Random Forest

O modelo Random Forest pode ser utilizado na mensuração das importâncias dos atributos de um conjunto de dados para a previsão do modelo. O grau de importância de

um atributo é medido em pontuação gini e é calculado durante o processo de montagem das árvores que compõe o algoritmo. Os atributos com melhor pontuação de gini são aqueles que, quando utilizados como condição de divisão de um nó, resultam em subconjuntos com menor impureza. No caso do random forest, a pontuação de gini é uma média dessa métrica para cada uma das árvores. A Figura 9 exibe os valores de gini das duas estações que mais contribuíram para a determinação das coordenadas durante os treinamentos dos modelos em cada etapa de treinamento. As colunas verdes representam as terceiras estações mais importantes para as previsões, que variam ao longo das etapas. É notável que as importâncias das estações 080605EE e 08060716 diminuem progressivamente, o que se explica pelo aumento na quantidade de mensagens. A estação 080605EE teve sua maior importância durante o treinamento com as mensagens armazenadas até o dia 08/01/2019, subconjunto que também foi utilizado para desenvolver os modelos com os melhores desempenhos tanto no kNN quanto na Random Forest.

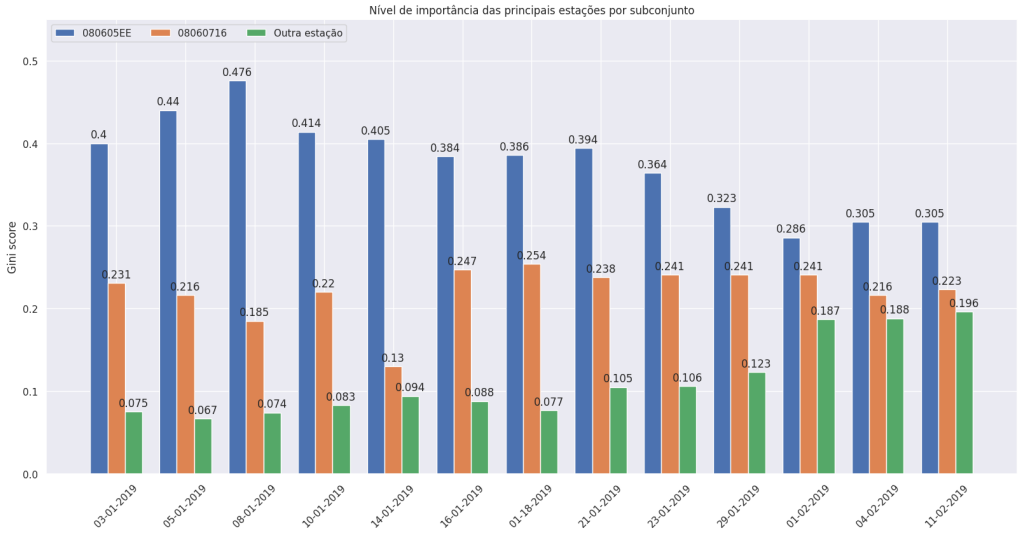


Figura 9. Estações mais importantes para as previsões dos modelos Random Forest

O gráfico exibido a cima mostra que as estações 080605EE e 08060716 apresentaram importâncias em uma proporção muito maior do que as demais estações durante as primeiras dez rodadas de treinamento. A ausência de um equilíbrio entre as importâncias das estações no decorrer das rodadas de treinamento evidencia a esparsividade do conjunto de dados, observação antes feita por [Anagnostopoulos and Kalousis 2019].

5.3. Rede neural RBF

Nesta subseção, a capacidade de redes neurais RBF em realizar previsões de coordenadas com o conjunto de dados utilizado é avaliada. Cada hiperpâmetro foi ajustado de forma isolada dos demais, para assim verificar seu impacto na mudança das previsões do algoritmo. O desenvolvimento dos modelos preditivos foram realizados com o intuito de apenas apresentar exemplos de seus desempenhos. Por essa razão, uma busca exaustiva de hiperparâmetros não foi feita.

As melhores quantidades de unidades de ativação encontradas foram 300, 320 e 350, juntamente com as larguras de banda 0,7 e 0,75. Os resultados dos testes com os

modelos desenvolvidos com redes neurais RBF estão exibidos na tabela 4. Épocas são o números de iterações do processo de aprendizado da rede, a quantidade de vezes em que o algoritmo recebeu os dados de treino.*Mean Absolute Error*, ou erro absoluto médio, foi a função de custo que apresentou melhores resultados nos primeiros testes e foi utilizada para todos os modelos, juntamente com o otimizador Adam, com taxa de aprendizado de 0.0000125.

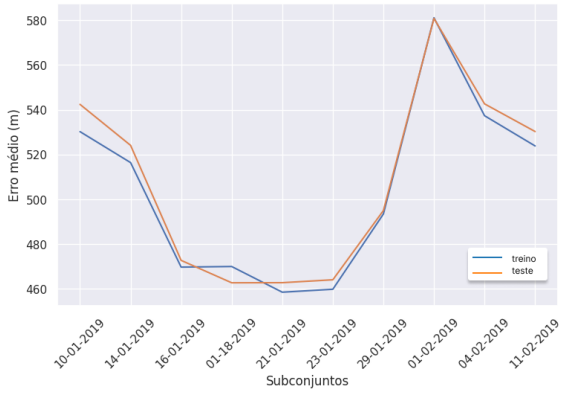
Intervalos	Largura de banda	Q. Unidades	Épocas	R²	Erro médio	Erro mediano
03/01/2019	0.75	350	1000	-419,9842	29969,32 m	22260,16 m
05/01/2019	0.75	350	1000	-45,3122	10676,85 m	7739,91 m
08/01/2019	0.75	350	1000	-0,5027	1785,66 m	1270,53 m
10/01/2019	0.75	350	1000	0,6878	542,42 m	369,21 m
14/01/2019	0.7	350	800	0,7800	524,12 m	388,51 m
16/01/2019	0.7	350	800	0,8113	472,74 m	370,38 m
18/01/2019	0.7	320	600	0,8176	462,73 m	348,99 m
21/01/2019	0.7	320	600	0,8158	462,78 m	347,85 m
23/01/2019	0.7	320	600	0,8188	464,07 m	343,26 m
29/01/2019	0.7	320	600	0,8210	495,00 m	367,56 m
01/02/2019	0.7	320	600	0,8153	580,81 m	422,51 m
04/02/2019	0.7	300	400	0,8047	542,65 m	389,49 m
11/02/2019	0.7	300	400	0,8148	530,29 m	384,90 m

Tabela 4. Resultados das rodadas de treinamento do modelo RBF

Apesar do mal desempenho três primeiros subconjuntos de dados, as redes neurais RBF apresentaram resultados razoáveis a partir da quarta rodada de treinamento, em que foi obtido um valor de R² positivo pela primeira vez. Comparativamente com os outros algoritmos avaliados, a rede neural RBF pareceu a abordagem menos apropriada para o problema de regressão, com melhor desempenho apresentado pela rede neural treinada com os dados enviados até o dia 18/01/2019, conforme apresentado na Tabela 5, onde é exibido os resultados das redes neurais. Os gráficos das Figuras 10(a), 10(b) e 10(c) mostram pouca distância entre as métricas observadas para com os conjuntos de treino e teste, comportamento que também difere dos demais algoritmos. Os gráficos exibem os resultados obtidos a partir do subconjunto das mensagens coletadas até o dia 10/01/2019 por uma questão de clareza, pois, os resultados dos três primeiros subconjuntos possuem uma magnitude diferente dos demais, o que dificultaria a visualização das curvas caso estivessem presentes nos gráficos.

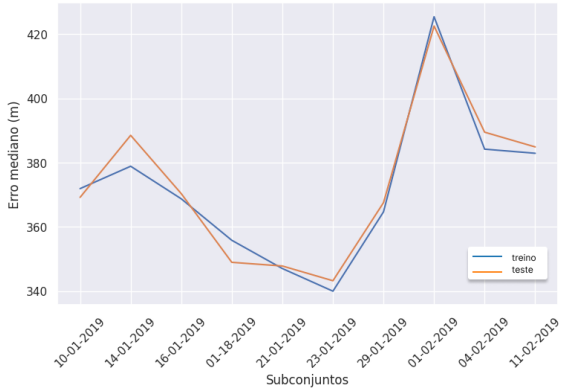
Intervalo	Algoritmo	Resultado
08/01/2019	Random Forest	301.34 m
08/01/2019	KNN	307.97 m
18/01/2019	Rede neural RBF	462,73 m

Tabela 5. Melhores resultados



(a) Erros médios por subconjuntos - RBF

(b) R² por subconjuntos - RBF



(c) Erros medianos por subconjuntos - RBF

Figura 10. Resultados do modelo RBF

6. Conclusão

No presente trabalho, uma comparação das performances dos algoritmos KNN, Random Forest e Rede neural RBF foi feita, no contexto de redes LoRaWAN e localização de seus dispositivos. Um conjunto de dados público de mensagens de dispositivos LoRa foi utilizado para o treinamento dos algoritmos. Foram realizados um ordenamento das mensagens de acordo com as suas datas de envio, e a separação do conjunto de dados em partições de acordo com a quantidade de dados, com o intuito de representar lotes de mensagens acumuladas a partir de uma data inicial. Em seguida, os algoritmos estudados foram treinados e avaliados com cada uma das partições. Os resultados obtidos foram reportados na seção 5, e a metodologia empregada para esse estudo é descrita na Subseção 4.3. As análises realizadas com o conjunto de dados foram exibidas na Subseção 4.1.

O melhor resultado foi obtido com o algoritmo Random Forest, treinado e avaliado com as mensagens acumuladas até o dia 08/01/2019. Um modelo que localiza dispositivos, em média, $301,34$ metros distantes da localização correta. O mesmo modelo também apresentou uma pontuação R^2 de $0,88$ e um erro mediano de $164,26$ metros em suas previsões. O algoritmo KNN também apresentou resultados similares aos do Random Forest, com erro médio de $307,97$ metros obtidos com as mesmas mensagens.

Para trabalhos futuros, a técnica fingerprinting pode ser replicada seguindo a metodologia descrita neste estudo, adotando uma estratégia de aprendizado profundo. Isso

visa aprimorar o entendimento acerca do potencial do conjunto de dados empregado na pesquisa. Adicionalmente, caso no futuro a comunidade científica tenha acesso a um conjunto de dados com um volume maior de mensagens, será possível realizar uma avaliação mais detalhada sobre o progresso da eficácia dos algoritmos de aprendizado de máquina desenvolvidos segundo a metodologia apresentado no presente trabalho.

Referências

Aernouts, M., Berkvens, R., Van Vlaenderen, K., and Weyn, M. (2018). Sigfox and lorawan datasets for fingerprint localization in large urban and rural areas. *Data*, 3(2).

Aggarwal, C. C. (2018). *Radial Basis Function Networks*, pages 217–233. Springer International Publishing, Cham.

Alliance, L. (Acesso em: 27 dez. 2023). About lora alliance®. *Disponível em: https://lora-alliance.org/about-lora-alliance/*.

Anagnostopoulos, G. G. and Kalousis, A. (2019). A reproducible comparison of rssi fingerprinting localization methods using lorawan. In *2019 16th Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6.

Anjum, M., Abdullah Khan, M., Hassan, S. A., Jung, H., and Dev, K. (2022). Analysis of time-weighted lora-based positioning using machine learning. *Computer Communications*, 193:266–278.

Centenaro, M., Vangelista, L., Zanella, A., and Zorzi, M. (2016). Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67.

Chaudhari, B. S. and Zennaro, M. (2020). 1 - introduction to low-power wide-area networks. In Chaudhari, B. S. and Zennaro, M., editors, *LPWAN Technologies for IoT and M2M Applications*, pages 1–13. Academic Press.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Géron, A. (2019a). Decision trees. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc.

Géron, A. (2019b). Ensemble learning and random forests. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc.

Géron, A. (2019c). The machine learning landscape. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc.

Janssen, T., Berkvens, R., and Weyn, M. (2020). *Comparing Machine Learning Algorithms for RSS-Based Localization in LPWAN*, pages 726–735.

LAB'S, L. (2016). A comprehensive look at low power, wide area networks. *White Papper*.

Lora Alliance. *LoRaWAN™ Backend Interfaces 1.0 Specification*.

Patel, D. and Won, M. (2017). Experimental study on low power wide area networks (lpwan) for mobile internet of things. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Torres-Sospedra, J., Montoliu, R., Trilles, S., Óscar Belmonte, and Huerta, J. (2015). Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems. *Expert Systems with Applications*, 42(23):9263–9278.

Vidnerova, P. (2019). Rbf-keras: an rbf layer for keras library. *Disponível em https://github.com/PetraVidnerova/rbf_keras*.

Yegin, A., Kramp, T., Dufour, P., Gupta, R., Soss, R., Hersent, O., Hunt, D., and Sornin, N. (2020). 3 - lorawan protocol: specifications, security, and capabilities. In Chaudhari, B. S. and Zennaro, M., editors, *LPWAN Technologies for IoT and M2M Applications*, pages 37–63. Academic Press.