UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

NOME

Douglas Tavares Ribeiro Paulino Silva

TÍTULO

Algoritmo de Localização *Outdoor* e *Indoor Fingerprinting* para Estações Móveis baseado em *LightGBM*

RECIFE

RESUMO

Ao longo dos últimos anos, tem observado-se um grande número de aplicações IoT e aplicativos para dispositivos móveis que utilizam o sistema de localização. Os serviços de localização apresentam inovação para o *marketing*, segurança, saúde pública e entre outras áreas. Umas das técnicas utilizadas para a localização de Estações Móveis (EMs) é a técnica de radiolocalização *Fingerprinting*. Essa técnica destaca-se por ter um algoritmo fácil de ser implementado e conseguir uma boa acurácia. A implementação do algoritmo consiste na geração do *Fingerprinting correlation database* (CDB) que possui células com o *fingerprint* das EMs para as Estações de Rádio Base (ERBs). Com a coleta do *Fingerprint* da EM é possível usar correspondência de padrões com o CDB para predizer a posição do usuário. O CDB geralmente é construído por modelos de propagação como Okumura-Hata, COST-231 e ECC-33, porém nos últimos anos os modelos de Aprendizagem de Máquina (AM) vem sendo mais utilizados, para superar as limitações dos modelos tradicionais de propagação como para ambientes urbanos. Alguns artigos da literatura utilizam o *fingerprinting* com *Support Vector Regression* (SVR), para predizer os RSSIs do CDB e obter uma melhor performance nos resultados. Este trabalho propõe uma nova solução utilizando *fingerprinting* com o *LightGBM* e será comparado com um artigo da literatura que utiliza o *fingerprinting* com SVR. O *LightGBM* propõe manter a mesma acurácia, com um tempo de treinamento e consumo de memória menor em relação aos outros algoritmos de AM. Além disso este algoritmo é ideal para treinar com um grande número de instâncias de dados e um grande número de *features*.

Palavras-chave: Sistemas de Localização *Outdoor* e *Indoor*, *Fingerprinting*, *Machine Learning*, Redes móveis, LightGBM.

ABSTRACT

Resumo em inglês.

Keywords: 3 to 5 keywords.

Sumário

1. Introdução

Ao longo dos anos, houve um aumento significativo no número de *smartphones*. A medida que os dispositivos móveis foram tornando-se mais populares, ocasionou o surgimento de uma diversidade de aplicativos para suprir as necessidades dos usuários. Para um melhor atendimento e qualidade do serviço prestado, alguns aplicativos de *smartphones* precisam da localização do usuário. Com isso, a localização de usuários em redes móveis e sem fio tornou-se muito importante para a inclusão digital. Governos dos países também utilizam a localização dos *smartphones* dos seus cidadãos para informá-los sobre tragédias ou conscientizá-los com alertas em carros de som. Foi o caso durante a pandemia do novo coronavírus (Sars-CoV-2), em que o Governo do Brasil, fez uma parceria com as operadoras de telefonia para monitorar como estavam as medidas de isolamento no país [1]. Porém, esse monitoramento requer uma alta precisão e acurácia para identificar se as pessoas estavam realmente dentro de casa, o que não se torna possível saber porque o erro médio dos dados de geolocalização das operadoras assumem valores no intervalo de 1,3 Km [2] a 32 Km para regiões mais afastadas das Estações Rádio Bases (ERBs), de acordo com a *Federal Communications Commission* (FCC) [3]. Assim, distâncias menores que 1,3 Km não seriam detectadas e apenas deslocamentos de raio de 32 Km seriam altamente confiáveis.

As técnicas de localização mais conhecidas atualmente são o *Global Positioning System* (GPS) e o RF *Fingerprinting*. Porém, o RF *Fingerprinting* destaca-se em relação ao GPS. Um exemplo é sobre as redes IoT que são caracterizadas por um grande número de objetos. Uma vez que aplicações IoT exigem uma coleta de dados com alta periodicidade, o consumo de bateria do GPS aumentaria consideravelmente, inviabilizando o seu uso para aplicações IoT [4]. Assim o RF *fingerprinting* é mais recomendado, porque é um algoritmo de fácil implementação e é mais acessível, pois não precisa de *hardware* extra e consome pouca bateria do dispositivo móvel. Consiste basicamente em duas etapas uma *offline* e outra *online*. Na fase offline, é construído um mapa de cobertura o *Fingerprinting correlation database* (CDB) que pode ser gerado por meio de coletas de campo ou por modelos de propagação, e.g., Okumura-Hata. Já na fase *online*, o *fingerprint* da estação móvel é coletado, e é usado para correspondência de padrões com o CDB para predizer a posição do usuário [5].

Apesar de algumas vantagens, também existem alguns problemas que podem prejudicar a técnica de radiolocalização. Uns dos principais problemas, é que o valor absoluto do RSSI pode variar de acordo com o fabricante do *smartphone* e do modelo de medição. Essa variação pode causar problemas na acurácia, das técnicas de localização baseadas em *fingerprinting* **[6]**. Pois na fase *off-line* tem um padrão e na fase *on-line* os valores no local são totalmente diferentes. Outro problema, seria na construção do CDB por meio de coletas do sinal emitido pela EM em ambientes *indoor*. Nesses ambientes, o acesso para a coleta de dados na maioria das vezes é privado, pois tratam-se de casas, fazendas e empresas que restringem o acesso. Portanto, impossibilita a medição com o *drive test* no ambiente. Apesar das restrições de acesso para os técnicos realizarem as medições, os usuários que estão no ambiente ainda querem os serviços de localização com qualidade, ou seja, com precisão e acurácia boas.

Para resolver os problemas citados. Observa-se que, o RF *Fingerprinting* é um vetor com várias *features* e cada *feature* pode ser obtida por meio de diferentes dados, e.g. distância e ângulos, que não precisam necessariamente constituir o *fingerprint*. Como esses dados são responsáveis pela formação das *features*, então existe uma relação entre eles. Assim, esses dados podem ser utilizados como entrada para treinar um algoritmo de Aprendizagem de Máquina e predizer as *features*, e.g., os RSSI (para cada ERB). Dessa forma, o problema de Localização dos dispositivos móveis seria modelado como um problema de Aprendizagem de Máquina (AM), como foi feito em [5] que utilizou o *Fingerprinting* com SVR. Essa modelagem do problema visa obter melhores resultados e suprir algumas deficiências em relação aos modelos tradicionais. Em síntese, ao inserir uma modelagem por AM é esperado um aumento na acurácia e uma melhor performance.

Portanto neste trabalho, é proposto utilizar um algoritmo de AM conhecido com LightGBM que será treinado com apenas dados *outdoor*. Depois do algoritmo ser treinado, ele será responsável por predizer os *RSSIs* (para cada ERB) nos ambientes *outdoor* e *indoor*. Em seguida, é utilizada a técnica de *Fingerprinting* para localizar o usuário. O algoritmo LightGBM é um *Gradient Boosting Decision Tree* (GBDT) altamente eficiente que utiliza duas novas técnicas: *Gradient-based One-Side Sampling* (GOSS) e *Exclusive Feature Bundling* (EFB). As duas técnicas GOSS e EFB lidam com um grande número de instâncias de dados e com um grande número de *features* respectivamente. O artigo do LightGBM mostra que o algoritmo consegue superar outras técnicas de AM em termos de velocidade

computacional e consumo de memória, mantendo a mesma acurácia para o mesmo conjunto de dados [7].

1.1. Objetivos

O objetivo geral é comparar a proposta deste trabalho, que é o *Fingerprinting* com o *LightGBM* (FP-LightGBM), com um artigo de referência que utiliza o *Fingerprinting* com *Support Vector Regression* (FP-SVR), como em [5]. O objetivo é treinar as duas técnicas de AM com apenas dados *outdoor* para prever a localização do usuário tanto em ambientes *outdoor* e *indoor*. Ao final, será comparado o erro médio em metros, consumo de memória, tempo de treinamento e tempo de teste. O erro em metros permite verificar se a acurácia da técnica de localização proposta continua igual ou melhor do que o algoritmo de referência. O objetivo é analisar como o treinamento dos algoritmos SVR e LightGBM apenas com dados *outdoor* afetou a acurácia para a predição da localização do usuário e destacar os pontos fortes em cada cenário. Será analisado também quais os benefícios que o *fingerprinting* com o LightGBM traz para as aplicações que utilizam técnicas de localização.

1.2. Estrutura do Documento

A estrutura do documento realizada foi: No Capítulo 2, é apresentado o referencial teórico com os conceitos básicos de localização de redes móveis, a técnica de RF *Fingerprinting* e os fundamentos teóricos sobre os algoritmos de Aprendizagem de máquina. No Capítulo 3, é apresentado os trabalhos relacionados (Estado da Arte), esse Capítulo contém os artigos da literatura que foram estudados para a elaboração desse trabalho. No Capítulo 4, aborda a metodologia, como foi o planejamento para a realização dos experimentos do trabalho proposto, como a descrição da base de dados utilizada, o algoritmo proposto e o experimento. No Capítulo 5, são discutidos os resultados obtidos dos experimentos e é feita uma análise comparativa entre os dois algoritmos propostos. Também é feita uma descrição do ambiente de execução do experimento. Por último, no Capítulo 6, são apresentados as conclusões e sugestões para trabalhos futuros.

2. Referencial Teórico

Neste Capítulo, são introduzidos alguns termos e conceitos utilizados ao longo deste trabalho. São descritos alguns trabalhos realizados na área de localização de redes móveis utilizando *Fingerprinting* e os fundamentos teóricos dos algoritmos de Aprendizagem de máquina utilizados nesse projeto, quais sejam, SVR e *LightGBM*. O Algoritmo *LightGBM* é o centro desse estudo.

2.1. Localização de Estações móveis

O serviço de localização mais conhecido atualmente é o *Global Positioning System* (GPS), muito utilizado por aplicativos como *Google Maps* [9], *Waze* [10] e *Foursquare* [11]. O GPS apresenta uma precisão baixa, acurácia média e frequência baixa, é mais recomendado para ambientes *outdoor*. Pois, em ambientes *indoor* o sinal sofre com interferências, devido aos materiais que constituem o ambiente como: tijolos, madeiras e vidros. Essas interferências também ocorrem em grandes centros urbanos devido a grande densidade de prédios próximos ou pelas condições climáticas adversas, e.g., céu nublado. As interferências causam ruídos no sinal de rádio emitido e recebido pela rede celular, causando perdas significativas no sinal. A precisão é afetada diretamente, o que dificulta a localização do usuário do ponto exato de onde ele está.

Várias propostas podem ser encontradas na literatura, mas algumas técnicas de localização tendem a ser imprecisas, principalmente devido à propagação de *multipath* e *non-line-of-sight* (NLoS) [5]. Para rastrear os usuários em ambientes *indoor,* alguns especialistas investem em tecnologias que utilizam o *hardware* adicional*,* o que causa um custo mais elevado na aplicação. Já outros utilizam o *software*, que torna a aplicação mais acessível.

A radiolocalização, utiliza *software* e sensores já presentes nos dispositivos móveis, é uma técnica de localização que faz uso de parâmetros presentes nos sinais de rádio para prover a localização do usuário. Esta técnica pode utilizar diferentes parâmetros, tais como: *Time of arrival* (ToA), *Angle of Arrival* (AoA), *Received Signal Strength Indicator* (RSSI) e *Time difference of Arrival* (TDoA). As técnicas de radiolocalização que destaca-se são: o

fingerprinting que é um vetor com várias *features* como: ToA, AoA, RSSI e TDoA que são usadas para correspondência de padrões para predizer a posição do usuário [5]. E a trilateração que estima a distância aproximada da estação móvel para cada uma das 3 ERBs mais próximas. As posições das ERBs são os pontos de referência, então é possível calcular a interseção dos raios das ERBs com a estação móvel. Em seguida, montar um sistema de equações e determinar a posição geográfica aproximada do usuário [12]. A técnica de radiolocalização utilizada neste trabalho é baseada em *Fingerprinting*, e será mais detalhada na próxima Subseção.

2.1.1 Técnica de radiolocalização baseada em *Fingerprinting*

A técnica de radiolocalização baseada em *fingerprinting* pode ser aplicada em qualquer rede sem fio, não necessita de *hardware* adicional ou modificações na infraestrutura [13]. As principais vantagens de performance são: boa acurácia, latência de detecção, uso eficiente da bateria e funciona em ambientes *non-line-of-sight* (NLoS). O *RF fingerprinting* consome pouca bateria mesmo enviando dados periodicamente, pois utiliza-se de alguns sensores do *smartphone* de baixa potência. Permite também, um compromisso (*trade-off*) entre acurácia e eficiência energética, o que não é possível no GPS [13], que pode ser definido dependendo das especificações da aplicação. O *fingerprint* possui 3 tipos principais na literatura que são: *visual fingerprint*, *motion fingerprint* e o *signal fingerprint*. Ainda existe o *hybrid fingerprint,* que é uma combinação desses 3 tipos mencionados para obter uma acurácia melhor do que os já citados. Apesar de existirem muitas técnicas de localização baseadas em *fingerprint*, essas técnicas têm componentes em comum, como apresentadas em [14], que são: *fingerprint* que é um vetor com sinais de rádio; CDB; servidor de localização; redução do espaço de busca dentro do CDB; e o algoritmo de *matching*. Estes componentes serão explicados ao longo deste Capítulo. Neste trabalho, será detalhado melhor como funciona o *signal fingerprint*, em especial sinais de radiofrequência, que é responsável pela captura de impressões digitais (*fingerprint*) que chegam do sinal de rádio do *smartphone*. Com esse sinal capturado é possível realizar a correspondência de padrões com o CDB, e então é possível identificar a localização do usuário [13]. Neste trabalho, para gerar o *fingerprint* é necessário coletar os valores dos indicadores de intensidade do sinal recebido (RSSI, *Received Signal Strength Indicator*) e calcular os atrasos de propagação (PD,

Propagation Delay) das múltiplas ERBs para o aparelho celular na região de interesse. Então, o *fingerprint* será a combinação dos RSSIs e PDs.

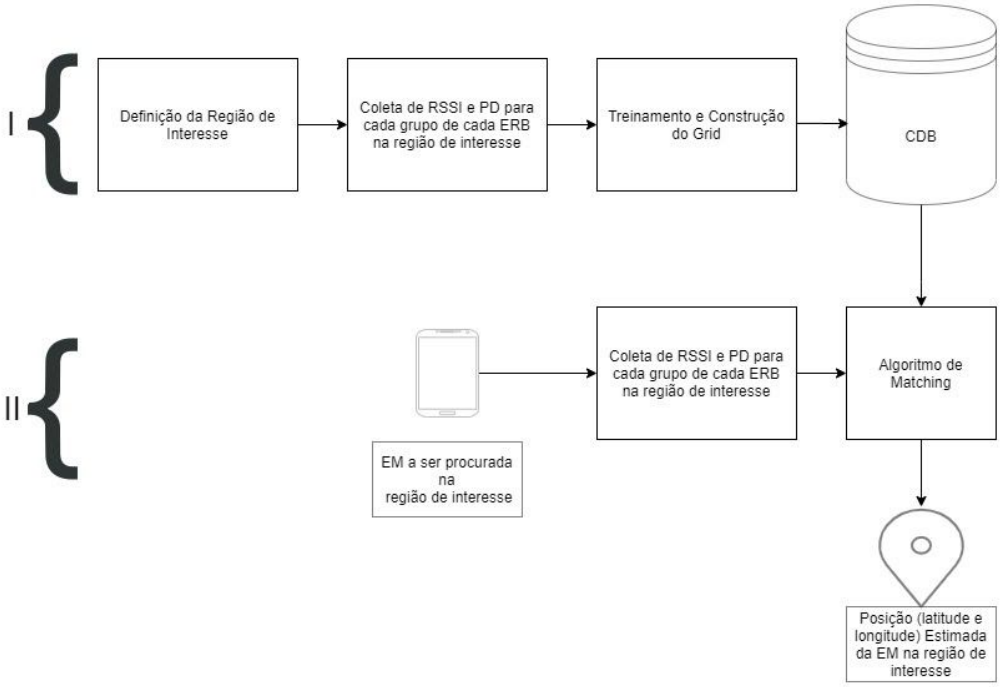


Figura 1 Diagrama da fase *on-line* e *off-line* da técnica de localização baseada em *fingerprinting* (Imagem adaptada de [15]).

O RSSI é um parâmetro que indica a potência do sinal de rádio recebido na EM da ERB. O seu valor é definido em dBm (decibéis relativos a um *miliwatt*) num intervalo de -100 à 0. Quanto mais o valor for próximo de 0, mais forte é o sinal, i.e., mais próximo a EM está da ERB. A unidade dBm é medida em escala logarítmica, ou seja, pequenas mudanças de dBm podem causar uma redução ou aumento elevado em relação a força do sinal anterior. É notável que, como é utilizada essa escala, caso sejam usados diferentes *scanners* para realizar a medição, e sabendo que o valor absoluto do RSSI pode variar de acordo com o fabricante. Isto pode implicar em mudanças bruscas na técnica de *fingerprinting* o que ocasiona resultados inferiores aos esperados.

Já o PD é um parâmetro de atraso de onda (ida e volta) da rede de celular, no qual o seu valor é definido por um número inteiro que varia de 0 à 56. Cada valor é mapeado por um intervalo de distância em passos de 234 *m* da EM para a ERB. Por exemplo, para as distâncias de [0, 234[*m* o valor de PD = 0, já para as distâncias de [234,468[*m* o valor PD = 1 e assim

sucessivamente. O último valor do PD é 56, e serve para todas as distâncias maiores que 13,1 *km* [16].

Assim, para construir um *fingerprint* é necessário a coleta de amostras de (RSSIs, PDs, latitude e longitude) na região de interesse da Estação Móvel (EM) para cada ERB. Esse formato foi proposto no trabalho [8] e é descrito a seguir, como $FP_{(x,y)}$:

$$FP_{(x,y)} = [P_l, \Gamma_l], \tag{1}$$

Onde $FP_{(x,y)}$ é o *fingerprint* do sinal gerado na região de interesse da Estação Móvel (EM) com coordenadas geográficas, em que x é a latitude e y é a longitude. P_l é o vetor que representa os valores de RSSI (em dBm) para a l th amostra.

$$P_l = \{P_{l,i}^{(u)}\},\ i = 1, \ldots, N;\ u = 1, \ldots, N_e, \tag{2}$$

$P_{l,i}^{(u)}$ é o valor do RSSI para u th ERB pertencente ao grupo i th. Por último, Γ_l é um conjunto de medições de PDs de tal modo que:

$$\Gamma_l = \{\gamma_{l,i}\},\ i = 1, \ldots, N, \tag{3}$$

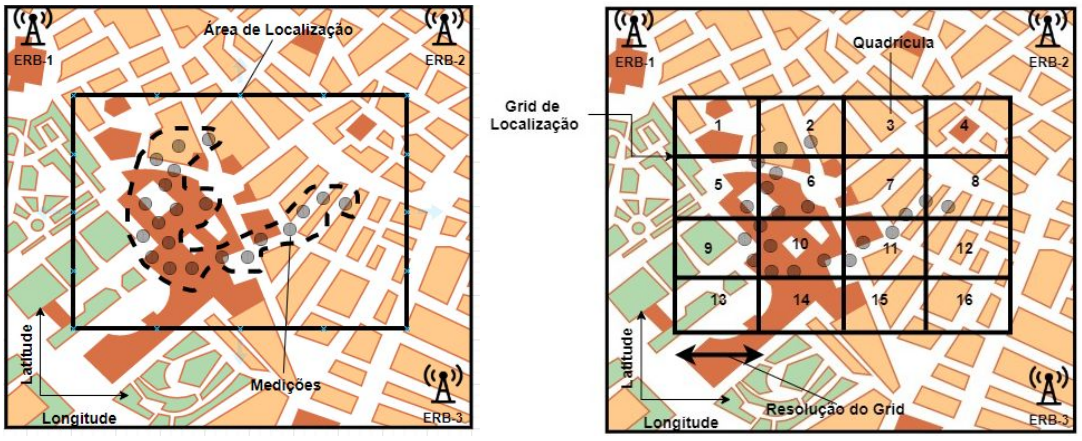
Onde $\gamma_{l,i}$ é a medição de PD para o i th grupos de ERBs. No caso deste trabalho, existem N = 3 grupos, e para cada grupo com N_e = 3 ERBS. Dessa forma, o *fingerprint* é construído.

Apesar de existir uma variedade de RF *fingerprintings*. Essa técnica de radiolocalização consiste de duas fases: a fase treinamento (*off-line*) e a fase de teste (*on-line*) [13]. A Figura 1 apresenta o diagrama de forma resumida de como funciona a técnica de localização baseada em RF *fingerprinting*. As duas fases serão mais detalhadas na próxima Subseção.

2.1.1.1 Fase de Treinamento (*off-line*)

O banco de dados, conhecido como *Fingerprint correlation database* (CDB) ou mapa de rádio, é construído na fase de treinamento (*off-line*). Pode ser construído a partir de medições com um *drive-test,* conforme a Figura 5 (a), ou com modelos de propagação, e.g., COST-231 ou ainda a partir de modelos de AM. Para realizar as medições é preciso de um

drive-test que é responsável por processar os sinais de rádio recebidos da ERB e de um GPS para salvar os dados das coordenadas geográficas. Na Figura 5 (a), é definida uma área de localização em forma retângular, dentro desta área em uma região com linha tracejada são feitas as medições com o *scanner* que são representadas por pontos em preto. Após as medições é preciso definir o *grid*. O *grid* pode ter dois formatos: O *grid* regular que é um quadrilátero que define a região de localização, e é dividido em células quadradas denominadas quadrículas [16], conforme a Figura 5 (b). Ou o *grid* irregular que pode assumir uma geometria mais adaptada ao método de localização. Neste trabalho, será utilizado o *grid* regular. O mapa de cobertura (*grid*) deve sempre associar os atributos do *fingerprint* medido e/ou predito com uma coordenada geográfica (latitude e longitude). O tamanho do quadrilátero pode ser definido pelas latitudes e longitudes mínimas e máximas da região de localização. Cada quadrícula do *grid* tem o mesmo tamanho, já a resolução (em metros) varia nos trabalhos, a resolução adotada neste trabalho é de um *grid* regular de 20 metros. A posição estimada da localização de cada quadrícula do *grid* é estimada sempre no centro. Portanto, a resolução do *grid* é um limitador da acurácia da técnica, pois é assumido que a EM está no centro da quadrícula [16].



(a) (b)

Figura 2 (a) Área de Localização com as Medições feitas em campo com *drive-test*. (b) *Grid* de Localização com respectiva resolução (Imagens adaptadas de [16]).

As principais vantagens de construir um CDB apenas com medições do *drive-test* na região de localização, é que os dados de *fingerprint* têm mais qualidade do que os de modelos de propagação. Visto que, os dados são coletados com o valor real naquele local e não são

estimados por modelos matemáticos. Um problema de realizar as medições, é que é preciso da permissão para coletar os dados em propriedades privadas, no qual existem restrições de acesso. Ao construir o CDB sem as medições das propriedades privadas na região, pode ocasionar no CDB incompleto. Outro fato, são as mudanças na região de localização, que também são um problema como: construção de um novo prédio, implantação/retirada de uma antena, ou uma árvore derrubada. Com essas mudanças o CDB fica desatualizado, então é necessário realizar uma manutenção para atualizar o CDB com novas medições [17]. Porém as medições exigem muito esforço dos técnicos. Em contrapartida, a construção do CDB por meio de modelos de propagação ou AM tende a ser mais fácil. Pois, apenas é necessário atualizar os parâmetros da rede em caso de implantação/retirada de uma antena ou fazer menos medições em caso da construção de um novo prédio, pois essas medições seriam utilizadas para treinar o modelo novamente.

Os principais modelos de propagação da literatura são: COST-231 [18] e ECC-33 [19]. Já os modelos de AM são: *k-nearest neighbor* (k-NN), Redes Neurais e SVM. Com esses modelos, é possível estimar os RSSIs para os ambientes em que não foi possível realizar as medições, devido às restrições. Assim, as quadrículas não contempladas com medições do *drive-test* serão preenchidas [20]. Ou seja, ao utilizar esses modelos o CDB fica completo, mesmo sem realizar as medições em toda região. Pois, os modelos são responsáveis por generalizar os RSSIs para cada quadrícula do *grid*. Em relação a acurácia, de acordo com os experimentos em [5] e [15], os modelos de AM apresentaram melhores resultados do que os modelos de propagação, principalmente em grandes centros urbanos. Existe também uma solução de construção do CDB híbrida, i.e., são feitas medições em campo e as partes incompletas são completadas com modelos de propagação e/ou AM. A solução híbrida pode gerar bons resultados dependendo da base de dados e dos modelos escolhidos. Neste trabalho, é utilizado a modelagem por Aprendizagem de Máquina.

2.1.1.2 Fase de Teste (*On-line*)

A fase de Teste ou *On-line*, consiste na coleta dos dados de RSSI e PD da EM na região de interesse. Em seguida, com o algoritmo de *matching* é possível calcular a correspondência de padrões dos parâmetros da EM com o CDB, que já foi construído na fase *off-line*. Por fim, é feito a predizer da posição geográfica (latitude e longitude) da EM procurada. A ideia principal é encontrar um *fingerprint* de referência no CDB que tenha a

maior similaridade possível com o *fingerprint* alvo da EM procurada [5]. Como o *fingerprint* de referência do CDB possui uma coordenada geográfica associada, que foi salva durante as medições. Então podemos predizer a coordenada geográfica da EM procurada, pois como os dois *fingerprints* possui uma similaridade, assumimos que as coordenadas geográfica são iguais. Para definir a maior similaridade é preciso calcular a equação da distância euclidiana, que geralmente é a métrica mais utilizada, e é descrita a seguir:

A distância euclidiana entre os pontos $X_i = (X_{i1}, X_{i2}, ..., X_{im})$ e $X_j = (X_{j1}, X_{j2}, ..., X_{jm})$, num espaço euclidiano n-dimensional (n é o número de *features* do *dataset*), é definido como $d(X_i, X_j)$, onde:

$$d(X_i, X_j) = \sqrt{\sum_{r=1}^{m} (X_{ir} - X_{jr})^2} \tag{4}$$

Os pontos X_i e X_j possuem os sinais de rádio, no qual X_i possui o *fingerprint* alvo da EM procurada e X_j possui o *fingerprint* de referência do CDB. Essa equação é utilizada fixando o X_i e alterando a cada interação o X_j. Isto é feito, pois estamos em busca do X_i e X_j com maior similaridade possível. As *features* utilizadas para o cálculo de similaridade entre o *fingerprint* de referência e o *fingerprint* alvo serão apenas por meio dos RSSIs e PDs.

Na fase *on-line*, à medida que a área de localização aumenta ou a resolução do mapa de cobertura diminui, em ambos os casos a quantidade de quadrículas aumenta. Desse modo, as consultas no CDB tendem a demorar e exigir um aumento de poder computacional. Com o aumento das quadrículas do *grid* de localização é necessário utilizar técnicas de filtragem de *fingerprint*. Desse modo, é possível reduzir o espaço de busca dentro do CDB. Com um espaço de busca menor o tempo de busca e de processamento também diminui. Existem também outras técnicas como: algoritmo genético (GA, *genetic algorithms*) [14], Filtragem de CDB (CDB *filtering*) [14] e avanço de tempo (*timing advanced-based approaches*) [21]. Neste trabalho será realizado a última técnica de filtragem citada, pois ela é responsável por reduzir o espaço de busca a partir de um filtro por PDs. Essas técnicas de redução de espaço tendem a ser melhor utilizadas em aplicações reais, devido a uma grande quantidade de ERBs

na área de localização, em torno de 10.000 a 20.000. Desse modo, cada microssegundo que se ganha é importante, pois no final tem um grande impacto na busca.

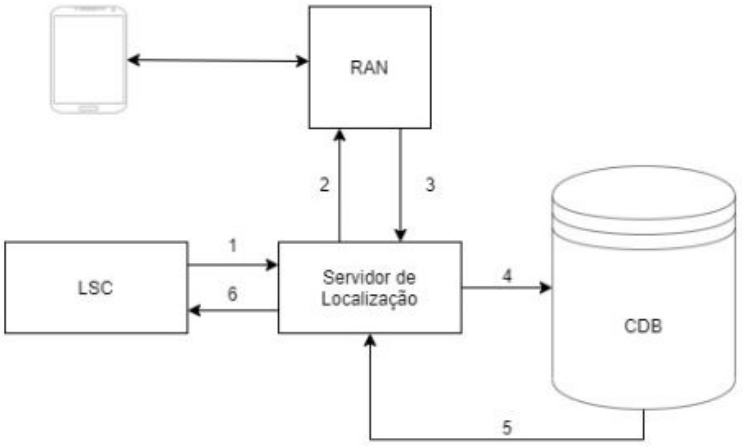


Figura 3 Diagrama simplificado do método de localização baseado em fingerprint (adaptado de [14]).

Na Figura 3, é apresentado num diagrama todas as etapas da fase *on-line*, às 6 etapas são descritas a seguir: A 1ª etapa, é do Cliente de Serviço de Localização (LSC, *Location Service Client*), que precisa da localização estimada da EM. O LSC é responsável por enviar uma requisição para o Servidor de Localização (*Location Server*). Na 2ª etapa, o Servidor de Localização faz uma requisição de medições de sinal da EM por meio da rede de acesso via rádio (RAN, *Radio Access network*). Em seguida, na 3ª etapa, RAN envia as medições de sinal coletas da EM para o Servidor de Localização. Com a medição recebida no servidor de localização é possível construir o *fingerprint* alvo da EM. Na 4ª etapa, com o *fingerprint* alvo o servidor de localização faz uma consulta no CDB, é retornado uma área de busca menor, no qual reduz o tempo de busca para a próxima etapa. Na 5ª etapa, o Servidor de Localização recebe os resultados da área reduzida, e é feita a busca do *fingerprint* alvo da EM no CDB com uma busca menor. Ao encontrar um *fingerprint* de referência com maior similaridade, então é obtida a posição estimada da EM. Na 6ª e última etapa, o servidor de localização envia uma resposta ao LSC com o resultado obtido.

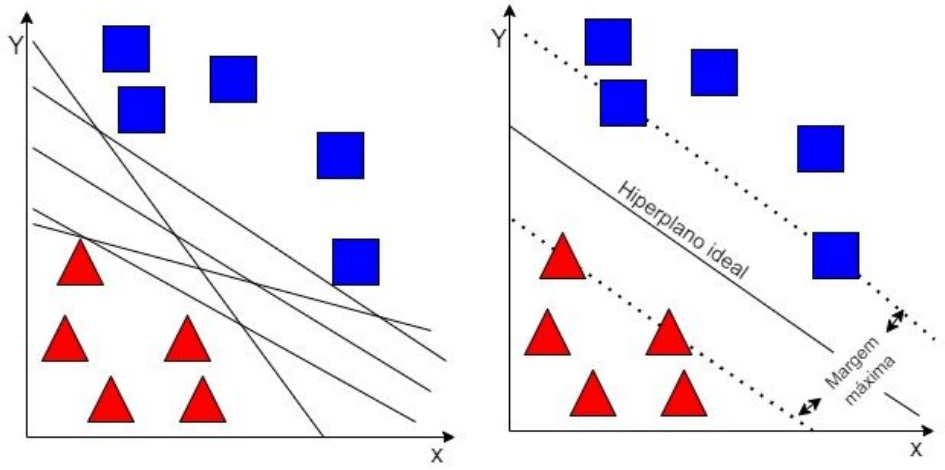
2.2 Aprendizagem de Máquina

Aprendizagem de Máquina (AM) é um subconjunto da Inteligência Artificial (IA), a principal definição é que AM é orientada a dados e os algoritmos de AM são modelos matemáticos baseados nestes dados. Existem três tipos de aprendizagem de máquina que são: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizagem por reforço.

Neste trabalho será utilizado a aprendizagem supervisionada, ou seja, o algoritmo precisa rotular os dados de entrada (variáveis independentes) com os dados de saída (variáveis dependentes) na fase de treinamento. Com esses dados rotulados, deseja-se encontrar uma função que seja capaz de predizer os novos rótulos desconhecidos. Na aprendizagem supervisionada pode-se ter duas formas de estimar os novos rótulos, por meio da classificação, que estima com base em um conjunto finito de valores discretos; ou pela regressão, que estima valores reais em um conjunto infinito. Existem vários algoritmos que funcionam das duas formas como *SVM*, *k-NN*, *LightGBM* etc. Nas próximas Subseções serão explicados os algoritmos de *LightGBM* e SVM com foco na forma de regressão.

2.2.1 Support Vector Machine (SVM)

As Máquinas de Vetores de Suporte (SVMs, *Support Vector Machines*) foram desenvolvidas na *AT&T Bell Laboratories* por Vapnik [22]. A técnica ganhou bastante atenção para diversos problemas na área de Aprendizagem de Máquina e reconhecimento de padrões. O SVM é um algoritmo de aprendizagem supervisionada, e é utilizado para problemas de classificação e regressão, no qual o último caso é mais conhecido como Regressão de Vetores de Suporte (SVR, *Support Vector Regression*) [23]. Este algoritmo serve para a detecção de *outliers* e além disso, resolve problemas linearmente separáveis e não-linearmente separáveis. Umas das vantagens do SVM, é que apresenta desempenho de generalização assim é possível evitar o *overfitting*. Em síntese, o *overfitting* é quando o classificador torne-se muito especializado memorizando apenas os dados de treinamento. Assim, o classificador não aprende e não apresenta resultados satisfatórios para um novo conjunto de dados.



(a) (b)

Figura 4 (a) Possíveis Hiperplanos que podem ser escolhidos para separar os rótulos de cada classe. (b) Hiperplano ótimo com a margem máxima entre os rótulos mais próximos de cada classe (Imagens adaptada de [22]).

O objetivo do algoritmo SVM, é encontrar um hiperplano em um espaço n-dimensional que separa os dados em diferentes classes. Na Figura 4 (a), são apresentados vários possíveis hiperplanos. Todavia, o ideal é encontrar a margem máxima que separa as duas classes, como na Figura 4 (b), pois assim encontramos o hiperplano ótimo. A margem é a distância do hiperplano até primeira instância de cada classe, cada instância que está em cima da linha pontilhada na Figura 4 (b) é conhecida como vetor de suporte.

2.2.1.1 Classificação de Padrões Linearmente Separáveis

A classificação linear é descrita como uma função real $f: X \subseteq \Re^N \rightarrow \Re^N$, e tem o objetivo de dividir os conjuntos em duas classes, de acordo com a entrada. Seja x uma entrada e $f(x) \geq 0$, então a instância é classificada como a classe positiva $(+1)$. Caso $f(x) < 0$, então a instância é classificada como a classe negativa (-1).

Um hiperplano pode ser representado pela seguinte equação:

$$f(x_i, w) = < x_i, w > + b \tag{5}$$

Em que x_i é o vetor de entrada e w é o vetor normal ao hiperplano, ambos $\in \Re^n$. O escalar b é o *bias* $e \in \Re$, já $< ., . >$ é o operador do produto interno entre os vetores. O cálculo do produto interno $< x_i, w >$ resulta em um número real que pode ser calculado da seguinte forma:

$$< x_i, w > \; = |x_i| \cdot |w| \cdot cos(\alpha) \tag{6}$$

Em que $||$ é a norma do vetor e α é o ângulo entre os vetores w e x. Para encontrar hiperplano ideal é necessário calcular a margem que é definida pela seguinte equação [24]:

$$margem \; = \; \frac{2}{||w||} \tag{7}$$

Da equação (7), podemos concluir que minimizando o valor da norma $||w||$, a margem tende a ser maximizada entre as duas classes. Com a margem calculada, o hiperplano ótimo é definido e a classificação dos rótulos é feita em duas classes. Supondo que todos os dados de treinamento seguiam as seguintes regras de (8) e (9) [24]. Seja x_i uma amostra dos dados de treinamento, então:

$$< x_i, w > \; + \; b \; \geq \; +1 \text{ , } x_i \text{ pertence a classe positiva (i.e., } y_i = \; +1 \text{)} \tag{8}$$

$$< x_i, w > \; + \; b \; \leq \; -1 \text{ , } x_i \text{ pertence a classe negativa (i.e., } y_i = \; -1 \text{)} \tag{9}$$

caso $< x_i, w > \; + \; b$, fique no limite de decisão, e.g., entre -1 e +1. Então x_i pode pertencer a uma das duas classes sem prejuízos. Para simplificar os cálculos as Equações (8) e (9) podem ser combinadas e formar a seguinte Inequação:

$$Y_i(x_i \cdot w \; + \; b) \; - \; 1 \; \geq \; 0 \;\; \forall_i \tag{10}$$

Uma observação é que a Inequação (10), não leva em consideração casos reais, pois não considera a presença de ruídos nos dados. Os ruídos podem violar as restrições de margem e causar mudanças na classificação. Dessa forma, pode ser adicionado uma variável de relaxamento ε que aumenta a sensibilidade do SVM, i.e., admite erros na classificação das classes. Porém, embora as variáveis de relaxamento permitam a sobreposição de distribuições de classes, essa estrutura ainda é sensível a *outliers*. Porque a penalidade para classificação incorreta aumenta linearmente com ε [26]. Ademais, pode afetar no número de vetores de suporte que são utilizados na construção da função de regressão. Assim a Inequação (10) com a variável de relaxamento é definida como:

$$Y_i(x_i \cdot w \; + \; b) \;\; \geq 1 \; - \varepsilon \;\; \forall_i \tag{11}$$

A variável de relaxamento ε pode ser adicionada também na Equação (7), porém é necessário adicionar um parâmetro de penalidade de termo de erro C, o qual controla o

comprometimento entre as margens grandes e pequenas violações de margem [26]. Desse modo, a Equação (7) torna-se:

$$margem = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\varepsilon_n \tag{12}$$

O parâmetro C é análogo a um coeficiente de regularização, porque ele controla o *trade-off* entre minimizar erros de treinamento e controlar a complexidade do modelo [25]. Para valores maiores de C, uma margem menor será aceita, se a função de decisão for melhor em classificar todos os pontos de treinamento corrente, porém implica no *overfit* do modelo. Já para valores menores de C, é preciso uma margem maior o que pode afetar a precisão do treinamento, causando mais erros de predição [27]. De acordo com (11) e (12), percebemos que ε e C definem a complexidade do modelo de treinamento do SVM. Por isso, esses dois parâmetros devem ser definidos de acordo com o problema e deve-se analisar quais os pontos fortes e fracos para obter bons resultados. Por fim, de acordo com os cálculos de [24]. Depois de treinar o algoritmo de SVM, na fase de teste para classificar um novo rótulo a classe correspondente é apenas necessário considerar x como:

$$sgn(w \cdot x + b) \tag{13}$$

2.2.1.2 Classificação de Padrões Não-Linearmente Separáveis

A grande parte dos problemas reais não são lineares, porém precisam ser tratados para ter solução. O SVM, trata os problemas não-linearmente separáveis de forma linearmente separáveis. Para isto, é necessário definir funções do tipo Φ que mapeiam o espaço de entrada para um novo espaço de características, ou seja, transformar x em $\Phi(x)$. Então, a Equação 13 também é alterada para adequar-se a essa transformação. Assim para classificar um novo rótulo a classe correspondente é:

$$sgn(w \cdot \Phi(x) + b) \tag{14}$$

O mapeamento com as funções Φ geram muitos cálculos computacionais e podem torna-se complexos. Para simplificar este mapeamento são utilizadas as funções *kernel*, pois são mais simples e não é necessário conhecer Φ para obter os cálculos e resultados. As

funções *Kernel* são responsáveis por calcular o produto interno entre dois pontos do espaço de entrada x_i e x_j no espaço de características, como na equação 14.

$$K(x_i, x_j) = < \Phi(x_i), \Phi(x_j) > \tag{14}$$

Tabela 1 Tipo de kernel com suas respectivas funções.

Tipo de Kernel	Função Kernel $K(x_i, x_j)$
Linear	$< x_i, x_j >$
Sigmoid	$tanh(\delta < x_i, x_j > + k)$
Polinomial	$(\delta < x_i, x_j > + k)^d$
RBF	$exp(-\delta \left\| x_i - x_j \right\|)^2$

As principais funções de Kernel são: *linear, sigmoid, polynomial* e *radial-basis function* (RBF), suas funções de Kernel podem ser vistas na Tabela 1 e para mais detalhes podem ser consultadas em [28]. Neste trabalho, será utilizada a função de Kernel RBF. Com a função de Kernel definida, precisamos realizar o ajuste de parâmetros de δ, ε e C. Assim, definimos quais os melhores valores para esses parâmetros para obter bons resultados. O método utilizado para o ajuste de parâmetros é o Grid search e será explicado melhor no Capítulo 5.

2.2.1.3 Support Vector Regression (SVR)

O SVM é mais conhecido por tratar de problemas de classificação, porém em 1996, foi proposto o *Support Vector Regression* (SVR), apresentado em [23], para tratar de problemas de regressão. Como visto nas seções anteriores, existem os problemas lineares separáveis e os não-lineares separáveis. O SVR também resolve os dois tipos de problemas seguindo o mesmo raciocínio do SVM. Porém, algumas mudanças são feitas no processo de treinamento do SVR em relação ao SVM.

$$min \ \tfrac{1}{2}||w||^2 \tag{15}$$

O problema é minimizar a Equação 15 para resolver o problema de otimização que no SVR tem 2 restrições que são diferentes da Inequação 11. Essas novas restrições serve para adaptar-se ao problema de regressão, no qual aproxima todos os pares de treinamento (x_i, y_i) com precisão até ε, então temos que:

$$y_i - \langle x_i, w \rangle - b \leq \varepsilon \quad \forall_i \tag{16}$$

$$\langle x_i, w \rangle + b - y_i \leq \varepsilon \quad \forall_i \tag{17}$$

O resto do processo é semelhante ao do SVM, até mesmo as funções de Kernel. Porém existem 3 tipos de algoritmos de SVR que são: ε-SVR [22], v-SVR [29] e ε-Bayesian [30]. Neste trabalho é utilizado o ε-SVR, no qual tem o objetivo de encontrar uma função que tenha no máximo desvio ε do alvo real y_i para todos os dados de treinamento. Enfim, o algoritmo ε-SVR não aceita qualquer desvio maior que ε [6]. Mais detalhes sobre SVM e SVR podem ser encontrados em [22-30].

2.2.2 LightGBM

O algoritmo LightGBM foi desenvolvido pela equipe da *Microsoft* em 2017, o código e documentação do algoritmo estão disponíveis em [31] e [32], respectivamente. O *LightGBM* Ele é conhecido como um framework que usa algoritmos de aprendizagem baseados em árvore de decisão, e também como um *Gradient Boosting Decision Tree* (GBDT) de alta eficiência. LightGBM é responsável por acelerar o processo de treinamento do GBDT convencional em até mais de 20 vezes, ao mesmo tempo que atinge quase a mesma precisão [7]. Este algoritmo está sendo muito utilizado recentemente em muitas competições de aprendizagem de máquina, tendo destacando-se dentre as principais soluções vencedoras do Kaggle [33]. Uns dos motivos deste algoritmo ser muito utilizado é que tem mostrado ser bem eficiente em várias etapas como: maior velocidade de treinamento e maior eficiência; menor uso de memória; melhor precisão; suporte à aprendizagem paralela e GPU; e é capaz de lidar com dados em grande escala [7]. A implementação do LightGBM é baseado em dois algoritmos que são: *Gradient-based One Side Sampling* (GOSS) e *Exclusive Feature Bundling* (EFB), que foram propostos para superar as limitações do algoritmo GBDT. As duas técnicas GOSS e EFB lidam com um grande número de instâncias de dados e com um grande número de *features* respectivamente. A grande maioria dos algoritmos de aprendizagem de árvore de decisão desenvolve árvores por *level (depth)-wise*, como *XGBoost*

que é uma implementação do GBDT. Na Figura 5, segue o exemplo de como esse algoritmo expande a árvore:

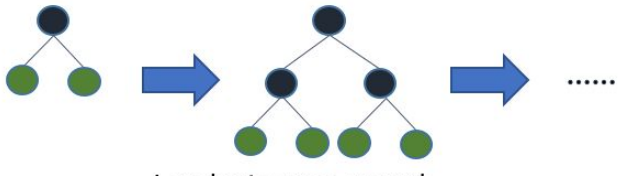


Figura 5 Representação do crescimento das árvores de decisão por *level (depth)-wise* [32].

As folhas em verdes podem ser expandidas, já os nós em preto não podem ser expandidos. Já o *LightGBM* produz árvores por *leaf-wise* (*best-first*), que pode ser visto na Figura 6.

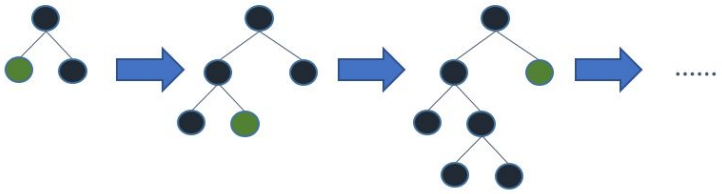


Figura 6 Representação do crescimento da árvore de decisão do LightGBM por *Leaf-wise* [32].

O crescimento da árvore é definido por uma regra específica, geralmente uma função de avaliação heurística $f(n)$, no qual essa função é responsável por escolher a folha com delta máximo de perda. A velocidade para a criação das árvores com *leaf-wise* provavelmente terá um desempenho melhor que *level-wise* [34], sendo assim faz com que o LightGBM seja mais rápido. Apesar da criação das árvores serem mais rápida, o *overfitting* pode acontecer principalmente quando a base de treinamento é pequena. Para resolver este problema o algoritmo LightGBM possui um parâmetro *max_depth*, o qual limita a profundidade da árvore. O LightGBM possui uma enorme quantidade de parâmetros, e a principal dificuldade é definir quais são os melhores parâmetros e ajustá-los para o *dataset*. Os parâmetros escolhidos serão aqueles que servem para acelerar o treinamento do algoritmo, aumentar a acurácia e lidar contra o *overfitting*, de acordo com a documentação. Os parâmetros que serão utilizados terão uma breve descrição nesta Seção. O primeiro é o *boosting_type*, o qual possui diferentes tipos de métodos de *Gradient Boosting* como: GBDT, DART e GOSS. Neste trabalho, será utilizado o GOSS. O *learning_rate*, o qual é recomendado um número bem pequeno quando temos um alto número de iterações (*num_iterations*), serve para uma melhor

acurácia; *num_leaves* determina o número máximo de folhas em uma árvore, esse valor deve ser menor ou igual à $2^{(max\_depth)}$, um valor maior que isso pode causar *overfitting*. Para aumentar a velocidade do treinamento, usamos os parâmetros o *feature_fraction* que seleciona uma porcentagem de features a cada interação para a construção das árvores. Já contra o *overfitting* podemos utilizar os parâmetros *min_data_in_leaf* que é o número mínimo de registros que uma folha pode ter, e *min_child_weight* que é o *hessian* de soma mínima em uma folha.

2.2.2.1 GBDT

O GBDT [35] é um algoritmo de AM que apresenta vantagens devido a sua eficiência, acurácia e interpretabilidade. Porém, atualmente este algoritmo está lidando com desafios da *big data*, principalmente na troca entre precisão e eficiência [7], o que torna as implementações mais demoradas. O GBDT cria árvores de decisão sequenciais e cada árvore é construída com base no erro residual (erros até a iteração atual) da árvore anterior. Então, as predições serão feitas pela soma de todas essas árvores [35]. Os erros são minimizados usando o método gradiente. A parte mais demorada do algoritmo, é a árvore de decisão encontrar os melhores pontos de divisão (*splits*). Os *splits* são os valores das *features* de treinamento, o qual definem como os dados são divididos nos nós da árvore. Os principais algoritmos que encontram os pontos de divisão (*split*) são: o algoritmo *pre-sorted* e o *histogram-based*. O algoritmo *pre-sorted* é simples e pode encontrar os pontos de divisão ideais, porém, é ineficiente na velocidade de treinamento e no consumo de memória [7]. Já o algoritmo *histogram-based* é mais eficiente no consumo de memória e na velocidade de treinamento, porém não possui soluções de otimização esparsas eficientes [7]. *XGBoost* [36] é um algoritmo que utiliza GBDT, e suporta o algoritmo *pre-sorted* e o *histogram-based*. Já o LightGBM usa *histogram-based*, e a construção do histograma é em torno de $O(\#dados * \#feature)$. O pseudo-código do *histogram-based* pode ser encontrado em [7]. Diante das limitações da técnica do GBDT foi criado o *LightGBM* que possui as duas novas técnicas GOSS e EBF, que serão descritos nas próximas Subseções.

2.2.2.2 GOSS

A nova técnica GOSS é responsável por encontrar um equilíbrio entre reduzir os número de instâncias e manter a acurácia da árvore de decisão [7]. Ademais, é uma implementação do GBDT mais leve e mais recente, porém pode causar *overfitting* quando o

dataset é pequeno. Para isto, o GOSS faz as amostragens das instâncias com base nos gradientes. As instâncias com gradientes pequenos são bem treinadas (erro de treinamento pequeno), e as instâncias com gradientes grandes são mal treinados (erro de treinamento grande). Então, o GOSS mantém todas as instâncias com gradientes grandes e realiza uma amostragem aleatória nas instâncias com gradientes pequenos [7]. Isto é feito para garantir que não altere a distribuição dos dados. A análise teórica e o pseudo-código da técnica GOSS podem ser consultado em [7] para mais detalhes.

2.2.2.3 EFB

O EFB é responsável por reduzir o número de *features* e é definido por *default* na implementação do *LightGBM*. O parâmetro que é responsável pelo EFB é chamado de *enable_bundle*, o valor default é true. O valor false, implica que EFB é desabilitado na construção do *dataset*. A redução de *features* torna-se necessária, pois o problema que ocorre é devido a grande dimensionalidade do conjunto de dados (dezenas a milhares de dimensões), que ocorre em várias aplicações que utilizam AM e são dificilmente tratáveis. Os dados de alta dimensão em sua maioria são muito esparsos [7]. Esta dispersão do espaço de *features* oferece a possibilidade de projetar uma abordagem sem perdas para reduzir o número de *features*, pois muitas *features* são mutuamente exclusivas, i.e., nunca assumem valores diferentes de zero simultaneamente [7]. Como foi citado, a construção do histograma é de $O(\#dados * \#feature)$, porém o *LightGBM* consegue agrupar as *features* que são mutuamente exclusivas em uma única *feature*. O processo de agrupamento das features é conhecido como *bundle*, no qual, $\#bundle << \#feature$. Assim, o tempo de construção é $O(\#dados * \#bundle)$, o que torna a construção do histograma mais rápida e acelera a aprendizagem da árvore. O EFB divide-se em duas partes: 1ª identificar as *features* que podem ser agrupadas e 2ª Utilizar um algoritmo para mesclar as *features*. A 1ª parte pode ser provada com uma redução do problema de colorir o grafo [37], apresentada no Alg. 3 de [7], visto que o problema de identificar as *features* que podem ser agrupadas é *NP-hard*. Já a 2ª parte é o apresentada no Alg. 4 de [7].

3. Trabalhos Relacionados (Estado da Arte)

O Problema de localizar o dispositivo móvel por meio de sinais de radiofrequência (RF) como RSSI, AOA, ToA e TDoA em uma área de interesse, vem sendo estudado e desenvolvido durante os últimos anos. Grande parte das técnicas de posicionamento sem fio foram desenvolvidas a partir do Relatório de 1996, feito pela FCC [38]. Neste relatório, a FCC determinou que qualquer ligação realizada para o número 911 (Número de emergência dos Estados Unidos), a EM deve enviar a sua localização para a central de emergência. E a acurácia de localização deve ser de 100 metros para 67% dos casos e 300 metros para 95% dos casos [39]. O *Fingerprinting* é uma técnica que apresenta bons resultados para ambientes de áreas urbanas (*outdoor*) e não precisar de *hardware* adicional ao sistema. Os trabalhos que utilizam o *fingerprinting* como [4 e 15], buscam um sistema de posicionamento móvel que forneça alta precisão de localização da EM. Isto demanda pouco atraso e um custo computacional baixo. Assim, para atender às regras determinadas pela FCC. Alguns trabalhos como [4, 7, 15], tratam o problema de localização como um problema de aprendizagem de máquina. Visando aumentar a performance da técnica de *fingerprinting*. Isto é possível devido a grande quantidade de sinais de RF e outras *features* relacionadas a EM e a BTS. Existem uma boa quantidade de algoritmos de AM que foram combinados com a técnica de localização *fingerprinting*. No entanto, os que mais destacam-se e são mais utilizados são o *k-nearest neighbors* (k-NN) [40,41] e *Support vector machine* (SVM) [15]. Estes algoritmos são utilizados devido aos seus bons resultados. Em [42], foi levando em conta a teoria de aprendizagem estatística, com a finalidade de fazer uma comparação de vários modelos de localização baseado em *Fingerprinting*. No artigo, foi concluído que o uso do SVM era o mais recomendado do que os outros algoritmos.

Em [5], o SVR foi comparado em relação aos modelos tradicionais de *fingerprinting* como: FP COST-231 e FP ECC-33. Bem como, foi realizado a medição de tempo consumindo em três etapas do algoritmo proposto FP SVR que são: treinamento do modelo, construção do CDB e consulta de 100 EMs no CDB (tempo de predição). Neste artigo, apesar do SVR apresentar uma boa acurácia, o treinamento do modelo apresentou ser 81 vezes mais lento em relação aos outros dois. Porém, para a etapa de construção do CDB apresentou uma performance um pouco melhor, já a consulta de 100 EMs no CDB teve resultados similares

com as outras técnicas. Em [15], foram comparados dois algoritmos de AM o k-NN/ST e SVR/ST, porém apenas foi realizada a comparação dos tempos de treinamento e predição das duas técnicas. O k-NN apresentou ser mais veloz no treinamento do que o SVR/ST, tanto do *target* de Latitude quanto de Longitude, todavia o SVR/ST apresentou resultados melhores no tempo de predição para ambos os *targets*. Ao utilizar algoritmos de AM, sempre é necessário utilizar o ajuste de parâmetros para alcançar bons resultados. Estes ajustes são geralmente feitos manualmente por meio de tentativas e erros (testes empíricos). Porém, para um algoritmo que possui muitos parâmetros como o *LightGBM* fazer ajustes manuais pode torna-se um problema, pois leva muito tempo. Então, para alcançar bons resultados e não perder muito tempo os artigos de AM utilizam um algoritmo de otimização, como o *Grid search*. Este algoritmo também será utilizado neste trabalho.

O SVR é um algoritmo que apresenta poucos parâmetros para a modelagem do problema, isso implica que ele faz uma quantidade menor de cálculos para treinar o modelo. Além disso, o SVR na função de decisão utiliza os vetores de suporte, que são um subconjunto de pontos do treinamento. Em síntese, isto faz com que o algoritmo seja eficaz em termos de memória. Com isso, vemos que o SVR apresenta ser mais lento na fase de treinamento, apesar de obter uma boa acurácia e ser eficiente em termos de memória. Então, não é do meu conhecimento algum trabalho que apresente um algoritmo com bons resultados de predição e ainda assim apresente uma eficiência no tempo de treinamento e consumo de memória. A motivação deste trabalho, é que como o algoritmo *LightGBM* possa ser possível manter a acurácia da técnica e acelerar o processo de treinamento, melhorar o consumo de memória, construção do CDB e consulta ao CDB em relação ao SVR. O SVR será o *baseline* deste trabalho para que possamos comparar os resultados. As métricas de erro médio de localização, tempo e consumo de memória são as partes cruciais deste trabalho. Pois são elas que vão definir se este caminho de pesquisa com o *LightGBM* para a área de localização de dispositivos móveis podem trazer avanços significativos.

4. Metodologia

Neste Capítulo, é apresentada a metodologia do trabalho, no qual primeiro é feita a descrição da base de dados utilizada em [8]. Em seguida, são apresentados os detalhes do experimento com o algoritmo FP-*LightGBM*, o qual buscamos uma melhora de performance em comparação com o algoritmo FP-SVR. O experimento baseia-se em treinar ambos os algoritmos com dados *outdoor* e analisar com estes comporta-se diante de três cenários que são: *outdoor*, *indoor-outdoor* e *indoor*. Por último, é detalhado como será o experimento comparativo entre o FP-LightGBM e o FP-SVR.

4.1 Base de Dados Utilizada

A base de dados foi construída no artigo [8]. Neste artigo, é detalhado que as medições de propagação de ondas de rádio móvel foram coletadas de uma Rede GSM, com uma portadora na faixa de 1,8 GHz de uma rede móvel de terceira geração (3G), que utiliza a interface aérea W-CDMA. O ambiente urbano escolhido foi na cidade de Recife-PE, mais precisamente no bairro da cidade universitária no campus da UFPE, com uma área aproximadamente de 1,6 km^2 . Nessa área existem 3 *sites* na região de interesse, Figura 8, que são os locais onde são instaladas as ERBs. Cada *site* possui 3 ERBs, o que resulta em 9 ERBs no total. As ERBs são definidas a partir do ângulo de azimute das antenas, que possuem três ângulos: 0°, 120° e 240°. As medições foram realizadas por meio de um equipamento *scanner,* DRT4301A fabricado pela DRT (*Digital Receiver Technology*) [8], para obter os sinais de rádio recebidos pelo equipamento. Ao todo, foram realizadas 9.679 medições, que podem ser vistas na Figura 8, das quais 3.064 e 6.615 medições foram coletadas em ambientes *outdoor* e *indoor*, respectivamente. Vale ressaltar que essa base de dados contém medições repetidas, principalmente as medições *indoor*. Foram realizadas várias medições *indoor*, pois esse ambiente possui mais ruídos que podem prejudicar as predições. E também porque em [8] a técnica foi apenas treinada com dados *outdoor*. O ambiente *indoor* escolhido foi a área interna do Centro de Informática da UFPE. Cada amostra compõe uma coordenada geográfica (latitude e longitude); os RSSIs para cada ERB; um conjunto de medições de *propagation delay* (PD) e as diferenças entre eles; um *booleano* para informar se a amostra é *indoor (true)*

ou *outdoor (false)*; os deltas; um vetor de distâncias da EM para os *sites* e os ângulos com seu senos, cossenos e tangentes da EM para os *sites*.

Figura 7 Localização dos *sites* em preto e os pontos de localização onde foram coletadas as amostras de ondas de rádio pelo *scanner* em azul.

Para construir o mapa de cobertura CDB é preciso definir a região de interesse, i.e., é preciso delimitar uma área definida por um quadrilátero com as latitudes e longitudes mínimas e máximas, que foram extraídas das medições . Na Figura 9, observamos os quatros extremos do quadrilátero que são definidos pelas seguintes coordenadas geográficas, no qual o primeiro parâmetro é a latitude e o segundo é a longitude: (-8,05955;-34,95985), (-8,05955;-34,94511), (8,04642;-34,95985) e (8,04642;-34,94511).

Figura 8 Vista do satélite da região de interesse delimitado pelo quadrilátero no campus da UFPE, Cidade Universitária, Recife-PE.

4.2 Algoritmo Proposto

O algoritmo proposto neste trabalho é utilizar o algoritmo *LightGBM* aplicado à localização. Para utilizar este algoritmo, é necessário ajustar os parâmetros que foram descritos na Subseção 2.2.2. Para efeitos de comparação iremos utilizar o algoritmo SVR, como o método de referência (*baseline*). O SVR também precisa de ajuste de parâmetros que são apenas três: γ, C *e* ε, como foi feito em [5]. Em ambos os algoritmos iremos utilizar o método do *Grid Search,* para realizar uma comparação justa e obter o melhor resultado de cada algoritmo. O método proposto para ambos os algoritmos é dividir a base de dados em dois arquivos no formato (.csv), visto que a base de dados possui 9679 medições *outdoor* e *indoor*. Do mesmo modo que foi proposto em [8], para o treinamento dos algoritmos de AM será utilizado 3.064 medições *outdoor,* sendo 80% dos dados para o conjunto de treinamento e os 20% restantes para o conjunto de teste. Já das 6.615 medições *indoor* serão utilizadas 459 medições para teste, que servem para avaliar a acurácia da técnica de localização das EMs para ambientes *indoor*. Assim, o 1º arquivo servirá para treinamento (*off-line*) dos algoritmos contendo apenas dados *outdoor* e o 2º arquivo para teste contendo dados *outdoor* e *indoor*. O arquivo de treinamento serve para ambos os algoritmos aprenderem com a base de dados. Neste trabalho, temos um total de 21 atributos de entrada (*features*), que são utilizados para

treinar o algoritmo proposto, e 9 atributos de saída (*target*) que são os RSSIs da EM para cada ERB. Deste modo, o algoritmo treinado é capaz de estimar uma função que receba o vetor de *features* e retorne uma saída, que neste caso é o RSSI. Como temos 9 ERBs neste trabalho e precisamos da predição do RSSI para cada ERB. Então, o LightGBM será treinado 9 vezes, pois o algoritmo só faz a predição de um *target* por treinamento. Segue uma descrição breve das *features* utilizadas neste trabalho, que são: a distância da estação móvel para cada *site* (um site possui três ERBs) que são dist_1, dist_2, dist_3; PDs: delay_1, delay_2, delay_3, delay_12, delay_13 e delay_23; além dos ângulos com seus senos, cossenos e tangentes (total de 12 *features*) da EM para cada *Site*. Com o treinamento realizado com as *features* do 1º arquivo e o aprendizado concluído. O algoritmo será responsável por receber como entrada o *grid* de localização da área de interesse com resolução de 20 m. Este *grid* possui todas as *features* citadas mais as coordenadas geográficas, porém sem os RSSIs. Pois, o RSSI (para cada ERB) será predito pelo Algoritmo. Assim é construído o mapa de cobertura do *Fingerprinting correlation database* (CDB). Cada linha do CDB será um *fingerprint* de referência. Essa linha vai conter os valores preditos dos 9 RSSIs; mais os PDs e as diferenças entre os PDs associando a uma coordenada geográfica do *grid*. Feito isto o mapa de rádio está completo devido a sua generalização, pois possui um *fingerprint* de referência até para as regiões que não foram medidas. Na fase de predição (*on-line),* será utilizado o 2º arquivo para ambos algoritmos. Este arquivo de teste, será dividido em mais 3 arquivos diferentes: o arquivo 2.1 apenas com dados *outdoor*, o arquivo 2.2 com dados *outdoor* e *indoor* e o arquivo 2.3 apenas com dados *indoor*. Cada *fingerprint* alvo é comparado com todas as células do *fingerprint* de referência em busca da célula mais similar. O algoritmo responsável pela comparação é o *matching* que utiliza a métrica de similaridade de distância Euclidiana de acordo com os RSSIs, PDs e diferenças de PDs da EM e do CDB. O resultado do *matching* é que a posição da EM que é célula mais similar com o mapa de rádio. Assim, podemos estimar a posição mais próxima do ponto onde o usuário está.

4.3 Experimento

O Algoritmo 1 tem uma descrição mais concisa do que foi explicado na Seção 4.2. Algumas partes dessa descrição foram baseadas na metodologia do trabalho de [5] que fez um experimento do *Fingerprinting* com SVR. Porém, foram feitas algumas modificações para

adaptar-se a este trabalho, visto que o objetivo é utilizar o *Fingerprinting* com *LightGBM* e treinar apenas com dados *outdoor*.

Algoritmo 1: Descrição da técnica proposta baseada em FP-LightGBM

---

1. Definir a área de interesse (*grid* de localização);
2. Coletar as medições do *scanner* na região de interesse apenas nos ambientes externos (*outdoor*);
3. Treinar o algoritmo *LightGBM* com dados *outdoor* para predizer o RSSI (para cada uma das ERBs);
4. Construir o mapa de cobertura (CDB);
5. Coletar as medições (RSSIs e PDs) da EM procurada para cada ERB, tanto em ambientes *outdoor* quanto em ambientes *indoor*;
6. Redução do espaço de busca no CDB por meio da filtragem em função dos PDs;
7. Estimar a posição da EM (latitude e longitude) de acordo com o ponto mais próximo no espaço de busca.

Ao implementar o Algoritmo 1 descrito nessa Seção, é preciso definir as métricas de desempenho para que possam ser feitas as comparações entre a técnica do FP-LightGBM com o FP-SVR. São definidas as principais métricas utilizadas:

1. **Erro médio de localização em metros:** É definido por meio da latitude e longitude preditas com a latitude e longitude reais. Para obter o resultado em metros utilizamos o PyRadioLoc [43] que recebe como entrada duas coordenadas geográficas e retorna a distância em metros. Em seguida, é feito a média de todas as amostras para obter o erro médio.
2. **Consumo de memória:** É definido de quantos MBytes o processo de execução do treinamento do algoritmo consome de memória.
3. **Tempo de treinamento:** É o tempo em segundos total para o algoritmo ser treinado e aprender como os dados de entrada relacionam-se com os dados de saída, i.e., tempo de construção do CDB.

4. **Tempo de teste:** É o tempo total que 100 amostras demoram para ser preditas pelo algoritmo, este tempo inclui a predição dos RSSIs dos algoritmos de AM e o tempo de busca no CDB.

Para comparar as técnicas de localização FP-*LightGBM* e FP-SVR. É definido o erro ε como a diferença, em metros, entre a posição predita e a posição real. Iremos utilizar *k-fold* com K=10 e a métrica RMSE, que será utilizado para avaliar a acurácia da técnica de localização proposta. O método de *K-Fold* [44] é uma validação cruzada que serve para avaliar a generalização do modelo de AM, de acordo com a base de dados. Já para o erro médio $\overline{\varepsilon}$ em metros ,definido na Equação 18 como:

$$\overline{\varepsilon} = \sqrt{\frac{1}{K} \sum_{j=1}^{K} \varepsilon_j} \quad \textbf{(18)}$$

em que ε_j é o erro médio para o j-ésimo *fold* de teste.

Este método divide o conjunto da base de dados em K grupos, e possui K iterações. Sendo que para cada iteração é utilizado K-1 grupos para o treinamento e 1 *fold* para teste. O *fold* de teste não pode ser repetido na próxima interação, i.e., outro *fold* precisa ser utilizado. Ao final, é feito uma média dos resultados de cada interação e a acurácia é melhor definida. O experimento deste trabalho consiste em realizar todas as etapas do Algoritmo 1, todavia a Etapa 3 tem mais ênfase, pois é o núcleo de todo o experimento e é a partir dessa etapa que podemos avaliar as métricas descritas neste trabalho. As etapas 3 e 4 consistem na fase *off-line*, já as etapas 5 e 6 consistem na fase *on-line* da técnica de localização. Como neste trabalho iremos fazer uma comparação para avaliar a eficiência do algoritmo proposto. Precisamos que o algoritmo SVR passe pelas mesmas etapas do LightGBM, pois assim podemos fazer as comparações justas. Então, para o caso da técnica de FP-SVR, apenas é necessário substituir na Etapa 3 do Algoritmo 1 o LightGBM pelo algoritmo SVR. E assim teremos ambos algoritmos treinandos com apenas dados *outdoor* que podem predizer a localização da EM tanto para ambientes *outdoor* quanto *indoor*. Ambos os algoritmos utilizam o *Grid Search* para otimizar os resultados por meio de ajuste de parâmetros. O *Grid Search* é responsável por definir um conjunto de valores para cada parâmetro, após isso são feitas múltiplas combinações de parâmetros e também é utilizado um *Cross-Validation* interno, com k = 3. Por fim, o *Grid Search* informa os melhores valores dos parâmetros para o

conjunto da base de dados. Para utilizar o *Grid search* no algoritmo LightGBM foi definido alguns parâmetros padrões que não seriam mudados. Foi o caso do *boosting_type* que foi definido como GOSS e o metric como MSE. Já o *learning_rate* foi definido como 0,1 e o número de interações como 200 no início, porém estes valores foram definidos para obter os outros parâmetros mais rápido. Visto que, quando estes dois parâmetros possuem os valores corretos, eles são responsáveis diretamente pelo tempo de aprendizado do algoritmo. O que resultaria em uma demora a mais no *Grid search* para obter os mesmo resultados. Ao obter os resultados dos outros parâmetros, depois só precisamos diminuir o *learning_rate* para 0,005 e aumentar o número de interações (*num_iterations*) para 400. Pois, o *LightGBM* tende a obter melhores resultados com um número baixo de *learning_rate* e um valor alto de número de interações. Feito isso, os outros parâmetros foram avaliados no *Grid Search*: como o *max_depth* com 6 valores (5 a 10), o *num_leaves* com 7 valores (2^3 a 2^9); *min_data_in_leaf* com 6 valores (1; 11; 21; 31; 41; 51); o *feature_fraction* com 5 valores (0,6; 0,7; 0,8; 0,9; 1,0); *lambda_l1* e *lambda_l2* cada um com 5 valores (8,0; 8,5; 9,0; 9,5; 10,0); *min_split_gain* com 5 valores (5 a 10); *min_child_weight* com 4 valores (11; 21; 31; 41); *top_rate* com 5 valores (0,50; 0,60; 0,70; 0,80; 0,90) e *other_rate* com 5 valores (0,05; 0,06; 0,07; 0,08; 0,09; 0,10). Depois de obter os resultados dos parâmetros pelo o *Grid search* é dado início a Etapa 3. Nesta etapa com os seus parâmetros já ajustados, é feito o treinamento do Algoritmo *LightGBM* que recebe como entrada as medições *outdoor*, em formato de *fingerprint.*

Além disso, é medido o tempo de treinamento e consumo de memória nesta etapa para ser avaliado nos resultados do Capítulo 5. Na Etapa 4, é o processo de construção do CDB para toda a área de interesse, por meio da generalização do algoritmo. Para realizar esta construção os valores de RSSIs são preditos pelo LightGBM, em seguida é feito a junção dos RSSIs preditos mais os PDs que foram calculados associando a uma coordenada geográfica. Então, é criado o *fingerprint* de referência para cada ponto no *grid* de localização, assim o CDB está completo, inclusive para as áreas privadas. Na Etapa 5, é a coleta do *fingerprint alvo* da EM procurada para cada ERB. Ou seja, é feito a coleta de RSSIs e PDs de cada EM, tanto em ambientes *outdoor* quanto em ambientes *indoor*. Nesta etapa foram escolhidos 1200 amostradas das quais metade são dados *outdoor* e a outra metade dados indoor. Porém ainda foi feito outra divisão para avaliar os três cenários propostos neste trabalho. Então os dados ficaram divididos da seguinte forma 400 dados outdoor, 400 dados indoor e 400 dados

outdoor-indoor (200 dados outdoor e 200 dados indoor), que ficaram misturados. Na Etapa 6, é gerado o mapa de redução do CDB por meio da filtragem em função dos PDs. Como o mapa reduzido não é necessário fazer uma busca por similaridade em todo o CDB e sim apenas nos fingerprints de referências que possuem os mesmos PDs do fingerprint alvo. O cálculo da equação de similaridade será executado menos vezes em razão deste filtro. Na 7 Etapa, a localização geográfica de cada EM é predita e é salvo o tempo de busca e de teste para cada 100 EMs. Esses resultados serão discutidos no próximo Capítulo. Na Tabela 2, podemos observar os resultados do ajuste de parâmetros do *Grid search* para o algoritmo *LightGBM*.

Tabela 2 Resultados do ajuste de parâmetros do *Grid search* para o algoritmo *LightGBM*.

Parâmetros do *LightGBM*	Valores
max_depth	7
num_leaves	32
min_data_in_leaf	21
feature_fraction	1
lambda_l1	8,5
lambda_l2	10
min_split_gain	10
min_child_weight	41
top_rate	0,90
other_rate	0,07

Como já foi descrito nessa Seção, é proposto fazer uma comparação com o SVR, para isto ser realizado o Algoritmo 1 é executado novamente apenas alterando o algoritmo LightGBM pelo algoritmo SVR, na etapa 3. Todos os outros passos permanecem inalterados.

Antes de executar a Etapa 3, é preciso utilizar o *Grid search* também no SVR. O único parâmetro fixo é o kernel que utiliza o rbf. Já os valores dos outros parâmetros para o algoritmo SVR foram baseados em [5], com pequenas mudanças. Para γ foram testados 9 valores (2^{-5} a 2^{3}), 7 valores para ε (0,01; 0,05; 0,1; 0,5; 1; 2; 4) e 12 valores para C (2^{-2} a 2^{9}). Os tempos de treinamento, busca no CDB, tempo de teste e consumo de memória também serão salvos e discutidos no Capítulo 5. Na Tabela 3, podemos observar os resultados de ajuste de parâmetros do *Grid search* para o algoritmo *SVR*.

Tabela 3 Resultados do ajuste de parâmetros do *Grid search* para o algoritmo *SVR*.

Parâmetros do SVR	Valores
γ	8
ε	0,5
C	64

5. Resultados

Neste Capítulo, é detalhado as plataformas de codificação, as bibliotecas que ajudaram no desenvolvimento e o ambiente de execução do experimento. Em seguida é apresentado os resultados obtidos do Experimento da Seção 4.3. Em resumo, é discutindo os resultados da execução dos dois modelos FP-*Fingerprinting* e *FP-SVR*. Ou seja, serão feitas análises comparativas dos resultados dos dois algoritmos como: média de erros de localização em metros nos três cenários (*outdoor*, *outdoor-indoor* e *indoor*), consumo de memória, tempo de treinamento e tempo de teste. Por último, é discutido se atingimos o resultado esperado proposto neste trabalho.

5.1. Ambiente de execução do experimento

Os experimentos realizados na Seção 4.3 deste trabalho são simulações computacionais que foram realizadas por meio da linguagem de programação Python 3.7.3. Essa linguagem de programação tende a ser melhor para trabalhar com problemas de AM devido à disponibilidade de várias bibliotecas que facilitam tanto a análise dos dados, quanto a implementação dos algoritmos como: SVR e LightGBM. A biblioteca Scikit-learn 0.21.2, foi responsável por fornecer o modelo SVR e pelos métodos Grid Search e Cross-Validation que melhora performance ajustando os parâmetros e realiza a validação da acurácia dos algoritmos, respectivamente. A biblioteca LightGBM 2.3.1, foi utilizada para fornecer o modelo LigthGBM que é o algoritmo de estudo deste trabalho. Outras bibliotecas também ajudaram no desenvolvimento dos algoritmos, como a biblioteca Pandas 0.24.2 que foi utilizada para análise e manipulação de dados. Para os cálculos numéricos realizados neste trabalho foi utilizado a biblioteca Numpy 1.16.4. Já a biblioteca Seaborn 0.9.0, foi utilizada para a visualização de dados que permite a visualização de gráficos, e.g., histogramas e boxplots. Por último, também foi utilizada uma biblioteca *PyRadioLoc* [43] que auxilia com o cálculo de distâncias geográficas. Os códigos dos algoritmos foram desenvolvidos nas IDEs Jupyter Lab notebook e no editor Visual Studio Code. Para não haver perda de códigos, foi utilizado o Github para realizar o versionamento do código do projeto. Por último, este experimento foi executado no sistema operacional Windows 10 Pro, num notebook Dell de 16

GB de memória RAM, com um SSD de 256 GB, processador Intel Core i7 8ª geração e placa de vídeo Nvidia GeForce MX130.

5.2. Análise Comparativa

Para avaliar o desempenho da técnica proposta FP-*LightGBM,* neste trabalho precisamos compará-lo com o desempenho da técnica de referência *FP-SVR*. Como foi explicado na Seção 4.3, ambos os algoritmos precisaram passar pelas mesmas etapas, pois assim é feita uma comparação mais justa. Bem como, é possível discutir as vantagens e desvantagens de utilizar uma técnica ou outra para o problema de localização de usuários. As principais etapas que os dois algoritmos passaram foram: otimização de parâmetros com o *Grid search*, a implementação do Algoritmo 1 com o treinamento apenas com dados *outdoor* e o *cross-validation*. Assim, neste Capítulo é feito uma comparação dos resultados obtidos na Seção 4.3 de ambos algoritmos.

O primeiro a ser discutido é o erro ε em metros, que como foi explicado na Seção 4.3 é a diferença entre a posição predita e real em metros. Porém, como a fase de teste tratar-se de muitas amostras é feito uma análise estatística do erro ε. Que é realizada para o tipo de ambiente e a técnica de localização executada, que pode ser vista na Tabela 4. Como já foi explicado, temos 1.200 medições de teste, das quais metade são *outdoor* e outra metade são i*ndoor*, que foram divididas em três ambientes *Outdoor*, *Indoor* e *Indoor-Outdoor*. As principais análises são: o erro médio de localização, em metros, que é definido por $\bar{\varepsilon}$, o desvio padrão ε_σ, o erro máximo ε_{MAX} e o mínimo ε_{MIN}. Podemos observar na Tabela 4 que o *FP-LightGBM* apresentou um erro médio $\bar{\varepsilon}$ de 43,91 m no ambiente *Outdoor*; 60,65 m no *Indoor* e 49,08 metros no *Indoor-Outdoor*. Já o *FP-SVR* apresentou um erro médio $\bar{\varepsilon}$ de 48,76 m no ambiente *Outdoor*; 90,08 m no *Indoor* e 70,52 m no *Indoor-Outdoor*. De acordo com os dados que foram apresentados, podemos concluir que a técnica *FP-LightGBM* apresentou uma melhor acurácia em todos os ambientes propostos em relação ao *FP-SVR*. O maior destaque ocorreu nos ambientes *indoor* e *indoor-outdoor* com uma diferença de 29,43 e 21,44 metros, respectivamente. Nota-se também que o erro mínimo ε_{MIN} *indoor* na técnica *FP-LightGBM* é de 21,40 metros, cerca de 10,73 metros melhor em relação ao *baseline*. Para os outros dois ambientes a diferença do erro mínimo é muito pequena, cerca de centímetros, e

a técnica deste trabalho só perder no ambiente *outdoor* por uma diferença de 24 cm. Já para o erro máximo ε_{MAX} o *FP-LightGBM* apresentou valores bem menores em relação ao *baseline*, principalmente nos ambientes *outdoor* e *indoor*. Já no ambiente *indoor-outdoor* a técnica *FP-SVR* destaca-se, mas a diferença é pequena, cerca de 2,08 m.

Tabela 4 Erro médio, desvio padrão, erro máximo e erro mínimo de predição da posição da EM, em metros, para as técnicas FP-*Fingerprinting* e FP-SVR nos ambientes *Outdoor, Indoor* e *Indoor-Outdoor*.

Ambiente	**Técnica**	$\bar{\varepsilon}$	ε_σ	ε_{MIN}	ε_{MAX}
Indoor-Outdoor	FP-*LightGBM*	**49,08 m**	**31,32 m**	**1,17 m**	193,10 m
	FP-SVR	70,52 m	40,82 m	1,51 m	**191,02 m**
Outdoor	FP-*LightGBM*	**43,91 m**	**39,74 m**	0,90 m	**217,92 m**
	FP-SVR	48,76 m	43,44 m	**0,66 m**	257,88 m
Indoor	FP-*LightGBM*	**60,65 m**	24,64 m	**21,40 m**	**116,27 m**
	FP-SVR	90,08 m	**24,59 m**	32,13 m	137,37 m

A outra forma de analisar os resultados e o comportamento das duas técnicas, é por meio de histogramas. Pois, é possível ter mais detalhes das quantidades de amostras que obtiveram boas acurácias, podemos observar os histogramas das Figuras 9, 10 e 11. Nos histogramas, o eixo x representa o erro de predição de distância ε (em metros) e o eixo y a quantidade de amostras que têm o mesmo intervalo de ε. A análise do histograma mostra que em ambas as técnicas e nos três ambientes citados neste trabalho, as amostras seguem o padrão da FCC, como ter uma acurácia de até 100 metros para 67% dos casos e 300 metros para 95% dos casos [39]. Observa-se ainda, que a técnica proposta neste trabalho apresenta, em ambos os ambientes, uma quantidade maior de valores mais próximos da origem do que o *baseline.* Os valores concentram-se entre os erros de 0 à 100 m. A técnica *FP-SVR*, comporta-se do mesmo modo, porém apenas no ambiente *outdoor*. Nos ambientes *indoor* e *indoor-outdoor*, os valores de erros estão mais distribuídos entre 60 à 140 m. Assim, com

todos estes resultados apresentados, podemos concluir que a técnica proposta neste trabalho apresentou uma acurácia melhor do que o referencial teórico.

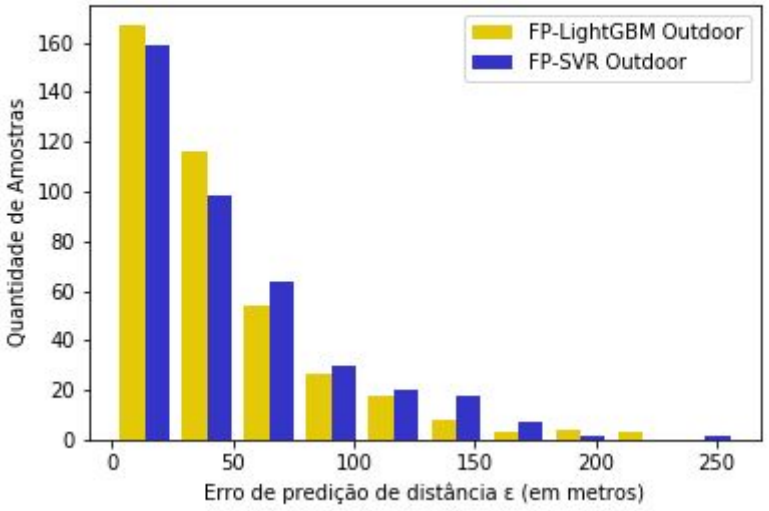


Figura 9 Histograma das duas técnicas de localização *FP-LightGBM* e *FP-SVR* no ambiente *Outdoor*.

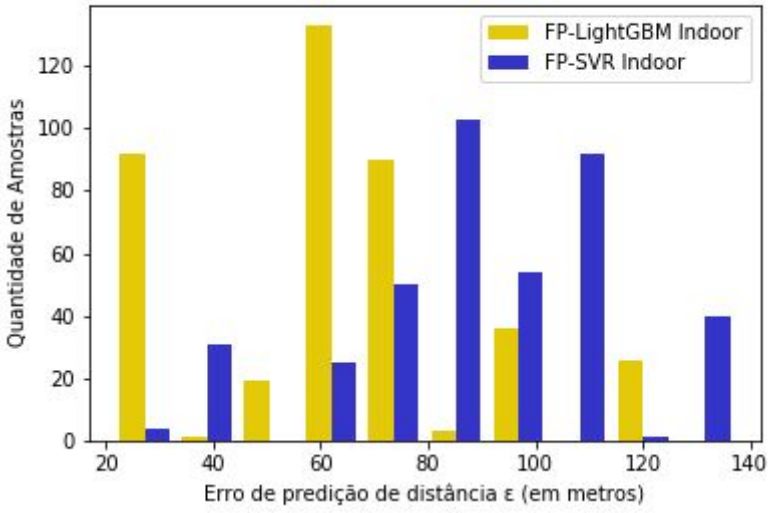


Figura 10 Histograma das duas técnicas de localização *FP-LightGBM* e *FP-SVR* no ambiente *Indoor*.

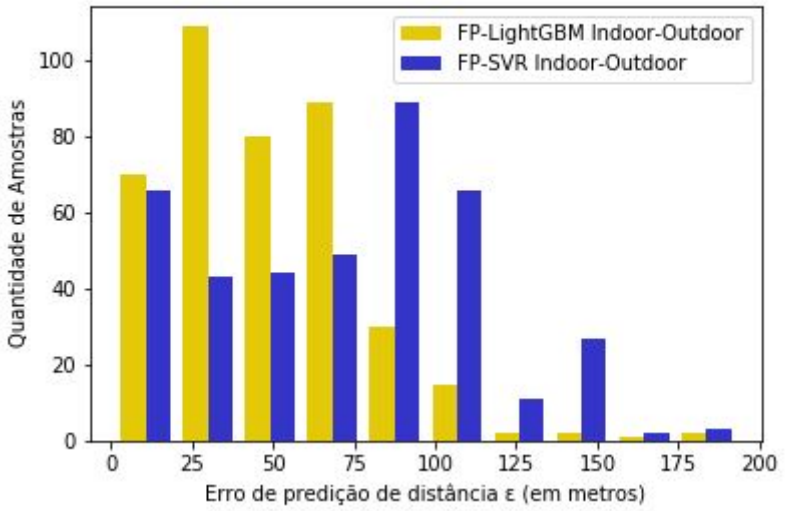


Figura 11 Histograma das duas técnicas de localização *FP-LightGBM* e *FP-SVR* no ambiente *Indoor-Outdoor*.

A outra comparação que foi realizada foi o custo computacional, em segundos, das fases de treinamento e de teste das técnicas de localização, que pode ser vista na Tabela 5. As definições dos tempos de treinamento e teste já foram apresentadas e podem ser consultadas na Seção 4.3. Durante cada experimento foi medido e salvo o tempo da execução dos processos das duas fases principais do *fingerprinting*. Todos os processos foram executados no mesmo ambiente de execução que foi apresentado na Seção 5.1 deste Capítulo. De acordo com a Tabela 5, observa-se que o tempo de treinamento do *FP-LightGBM* foi em média de 8,04 segundos, já o do *FP-SVR* foi de 14,58 segundos. Assim, o *FP-LightGBM* foi 44,85% mais rápido do que o *FP-SVR*. Já na fase de teste, com a medição do tempo médio de 100 EMs, os tempos foram iguais para as duas técnicas.

Tabela 5 - Custo computacional, em segundos, das fases de treinamento e de teste para as técnicas FP-*Fingerprinting* e FP-SVR.

Fase	**FP-*LightGBM***	**FP-SVR**
Treinamento	8,04 segundos	14,58 segundos
Teste (100 EMs)	1,77 segundos	1,77 segundos

Por fim, a última análise de desempenho, foi a comparação do consumo de memória, em MiB, de ambas as técnicas de localização na fase de treinamento. O consumo de memória como foi explicado na Seção 4.3 é medido a partir da execução do processo de treinamento, que se dá do início do aprendizado dos algoritmos até a construção do CDB. Na tabela 6, podemos observar o desempenho de ambas as técnicas, apesar da diferença ser de apenas 4,83 MiB a técnica *FP-LightGBM*, apresentou ser a mais eficiênte em memória em relação ao *FP-SVR*. Nas Figuras 12 e 13, podemos observar melhor o tempo e o consumo de memória durante a fase de treinamento.

Tabela 6 Consumo de memória, em MiB, na fase de treinamento das técnicas de localização FP-*Fingerprinting* e FP-SVR.

Técnica	**Memória**
FP-*LightGBM*	155,27 MiB
FP-*SVR*	160,10 MiB

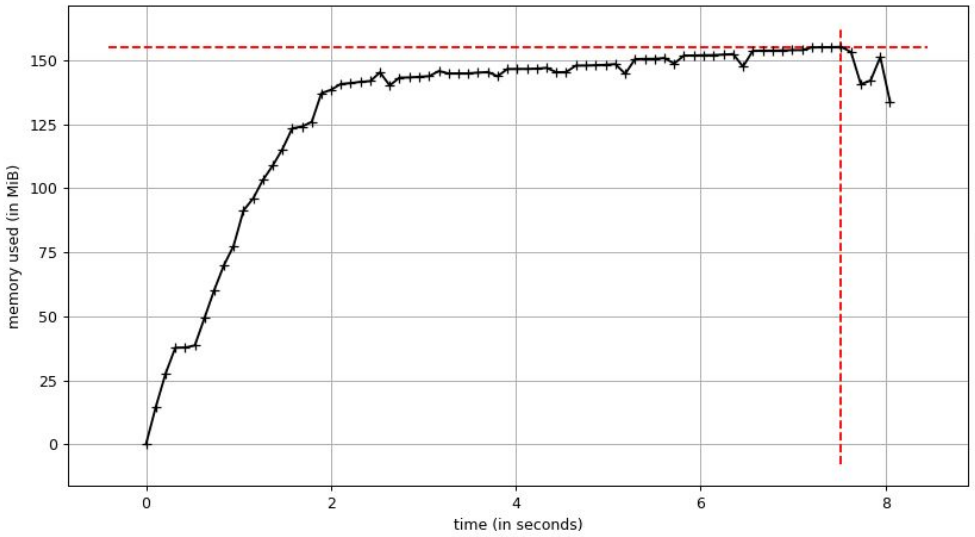


Figura 12 Gráfico que representa o tempo de treinamento, em segundos, e o consumo de memória, em MiB, da execução da técnica FP-LightGBM.

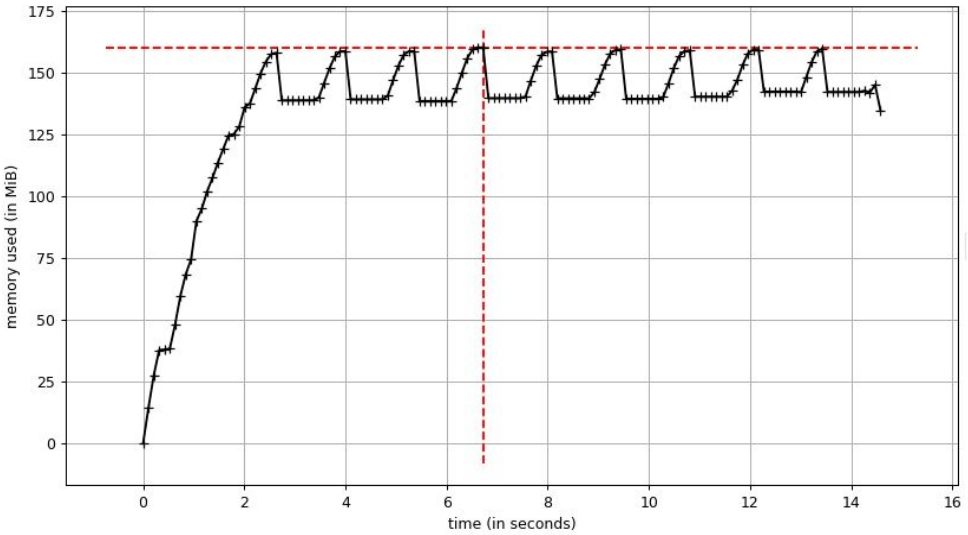


Figura 13 Gráfico que representa o tempo de treinamento, em segundos, e o consumo de memória, em MiB, da execução da técnica FP-SVR.

6. Conclusão

7. Bibliografia

[1] Governo vai usar dados de operadoras para monitorar aglomeração na pandemia, https://www1.folha.uol.com.br/mercado/2020/04/governo-vai-usar-dados-de-operadoras-para-monitorar-deslocamentos-na-pandemia.shtml, Acesso em: 02 de abril de 2020.

[2] Accurate Location Detection, https://transition.fcc.gov/pshs/911/Apps%20Wrkshp%2021 5/911_Help_SMS_WhitePaper0515.pdf Acesso em: 02 de abril de 2020.

[3] Cell tower data may be misleading, https://www.heraldcourier.com/opinion/cell-tower -data-may-be-misleading/article_f5e63928-7c76-5df2-8d38-b36388154a83.html, Acesso em: 02 de abril de 2020.

[4] GOLDONI, E, et al. Experimental data set analysis of RSSI-based indoor and outdoor localization in LoRa networks. Internet Technology Letters, [S.1], v.2, n.1, p.e 75, 2019 DOI: https://doi.org/10.1002/itl2.75

[5] TIMOTEO, Robson D.A.. SILVA, Lizandro N.. CUNHA, Daniel C.. CAVALCANTI, George D. C.. An approach using support vector regression for mobile location in cellular networks. Computer Networks 95, 2016.

[6] LUI, G. et al. Differences in RSSI readings made by different Wi-Fi chipsets: a limitation of wlan localization. In: INT. CONFERENCE ON LOCALIZATION AND GNSS (ICL - GNSS 2011). Proceedings… [S.1.: s.n.], 2011. p.53-57.

[7] KE, GUOLIN et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. 31st Conference on Neural Information Processing Systems (NIPS 2017) Pages 3149–3157, Long Beach, CA, USA.

[8] TIMOTEO, R. D.A.; CUNHA, D. C., A scalable fingerprint-based angle-of-arrival machine learning approach for cellular mobile radio localization, Computer Communications 157 (2020) 92–101. https://doi.org/10.1016/j.comcom.2020.04.014

[9] Google Maps, https://www.google.com.br/maps, Acesso em: 29 de março de 2020.

[10] Waze, https://www.waze.com/pt-BR/, Acesso em: 29 de março de 2020.

[11] FourSquare, https://pt.foursquare.com/, Acesso em: 29 de março de 2020.

[12] Savvides, A., Han, C. C., and Srivastava, M. B. (2001). Dynamic fine grained localization in ad-hoc sensor networks. In Proceedings of ACM Mobile Communications (MobiCom).

[13] A Survey of Fingerprint based Outdoor Localization

[14] J. Benikovsky, P. Brida, J. Machaj, . RF fingerprinting location techniques. Handbook of Position Location: Theory, Practice, and Advances, Wiley Online Library, 2011, pages 487–520, 2011.

[15] OLIVEIRA, L. L.. OLIVEIRA JR, L. A.. SILVA, G. W. A.. TIMOTEO, R. D. A.. CUNHA, D. C.. An RSS-based regression model for user equipment location in cellular networks using machine learning. Springer Science+Business Media, LLC, part of Springer Nature 2018. https://DOI.org/10.1007/s11276-018-1774-4

[16] Doutorado de Robson

[17] Heikki Laitinen, Jaakko Lahteenmaki, and Tero Nordstrom. Database correlation method for gsm location. In IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No. 01CH37202), volume 4, pages 2504–2508. IEEE, 2001.

[18] COST Action 231, “Digital mobile radio towards future generation systems, final report,” tech. rep., European Communities, EUR 18957, 1999.

[19] Electronic Communication Committee (ECC) within the European Conference of Postal and Telecommunications Administration (CEPT), “The analysis of the coexistence of FWA cells in the 3.4 - 3.8 GHz band,” tech. rep., ECC Report 33, May 2003.

[20] TIMOTEO, R. D.A.; CUNHA, D. C., A proposal for path loss prediction in urban environments using support vector regression. Proc. Advanced Int. Conf. Telecommun, 1-5, 2014.

[21] L. Klozar, P. Jan, Wireless network localization: optimization processing, in: Proceedings of the Seventh International Conference on Digital Telecommunications (ICDT 2012), Chamonix, FRA, 2012, pp. 45-49.

[22] V. Vapnik. "The Nature of statistical Learning Theory", pages 133–155. Springer, 1 edition, 1995.

[23] Vapnik, V., Golowich, S. E., & Smola, A. (1997). Support vector method for function approximation, regression estimation, and signal processing. In Advances in Neural Information Processing Systems 9 (pp. 281-287). Cambridge: MIT Press.

[24] C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". Data mining and knowledge discovery, 2(2):121–167, 1998.

[25] C. M. Bishop. "Pattern Recognition and Machine Learning", volume 53, pages 1689–1699. Springer, 2013.

[26] Léon Bottou and Chih-Jen Lin. Support vector machine solvers. Large scale kernel machines, 3(1):301–320, 2007.

[27] Yun Liu, Jie Lian, Michael R. Bartolacci, Qing-An Zeng. Density-Based Penalty Parameter Optimization on C-SVM, publicado em: 07/07/2014. https://doi.org/10.1155/2014/851814

[28] Ana Carolina Lorena, André Carlos Ponce de Leon Carvalho, et al. Introdução às máquinas de vetores suporte (support vector machines). 2003.

[29] A. J. Smola, Regression Estimation with support Vector Learning Machines, in: Master's dissertation, Technische Universit at Munchen, 1996.

[30] J. Gao, S. Gunn, C. Harris, M. Brown, A probabilistic framework for SVM regression and error bar estimation, Mach. Learn. 46 (2002) 71-89.

[31] Microsoft LightGBM, https://github.com/microsoft/LightGBM, Acesso em: 02 de abril de 2020.

[32] LightGBM documentation, https://lightgbm.readthedocs.io/en/latest/ Acesso em: 02 de abril de 2020.

[33] Kaggle, https://www.kaggle.com/, Acesso em: 06 de maio de 2020.

[34] Shi, H. (2007). Best-first Decision Tree Learning (Thesis, Master of Science (MSc)). The University of Waikato, Hamilton, New Zealand. Retrieved from https://hdl.handle.net/10289/2317

[35] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189-1232, 2001.

[36] Tiabqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pages 785-794. ACM, 2016.

[37] Tommy R Jensen and Bjarne Toft. Graph coloring problems, volume 39. John Wiley & Sons, 2011.

[38] Enhanced-911-WirelessServices, "http://www.fcc.gov/911/enhanced/."

[39] Federal Communications Commission. (1997). Revision of the Commission's rules to ensure compatibility with enhanced 911 emergency calling systems, CC Docket n. 94-102. Federal Communications Commission, Washington, DC.

[40] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RFbased user location and tracking system," in Proceedings of the 19th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00), vol. 2, pp. 775–784, IEEE, Tel Aviv, Israel, March 2000.

[41] E. S. Bhasker, S. W. Brown, and W. G. Griswold, "Employing user feedback for fast, accurate, low-maintenance geolocationing," in Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications (PerCom '04), pp. 111–120, March 2004.

[42] R. Battiti, M. Brunato, and A. Villani, “Statistical learning theory for location fingerprinting in wireless LANs,” Tech. Rep. DIT02-0086, Universita di Trento, Trento, Italy, 2002.

[43] Biblioteca para o cálculo de distâncias geográficas: PyRadioLoc https://github.com/timotrob/pyRadioLoc, Acesso: 09/04/2020