Universidade Federal de Pernambuco

Centro de Informática - Cin

Graduação em Engenharia da Computação

Uma Análise de Algoritmos de Aprendizagem de Máquina Aplicados em Técnicas de Localização Indoor para Diferentes Tipos de Smartphones

Larissa Lages de Oliveira

TRABALHO DE GRADUAÇÃO

Recife, Julho de 2017

Universidade Federal de Pernambuco

Centro de Informática - CIn

Graduação em Engenharia da Computação

Uma Análise de Algoritmos de Aprendizagem de Máquina Aplicados em Técnicas de Localização Indoor para Diferentes Tipos de Smartphones

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Aluna: Larissa Lages de Oliveira

Orientador: Daniel Carvalho da Cunha

Agradecimentos

Agradeço a minha família por todo o apoio e ajuda durante a minha graduação. Agradeço a todos meus amigos do CIn que sempre me ajudaram nos momentos mais difíceis da graduação. Agradeço aos meus colegas de trabalho do C.E.S.A.R por me darem dicas para este trabalho e me apoiarem. Um agradecimento especial a Victor Hazin e Rafael Bérgamo por acompanharem de perto este trabalho e me ajudarem a revisá-lo. Agradeço a Felipe Nunes por todas as dicas sobre aprendizagem de máquina, que foram muito importantes neste trabalho. Também agradeço a todos meus professores por me ajudarem a ser uma profissional cada vez melhor, especialmente a Edna Barros e Daniel Cunha, por terem me dado a oportunidade de trabalhar e aprender cada vez mais com eles durante minha graduação.

Resumo

Técnicas de localização *indoor* tem sido bastante estudadas nos últimos anos. Embora o GPS possa aproximar bem a posição do usuário em ambientes *outdoor*, em ambientes *indoor* a degradação do sinal é muito grande. Neste cenário, técnicas de *fingerprinting*, usando valores de intensidade do sinal de Wi-Fi (*received signal strength indicator*, RSSI), têm sido largamente utilizadas para localização de usuários por não precisarem de nenhum hardware adicional para fazer a localização. No que se refere às técnicas de *fingerprinting*, os algoritmos de aprendizagem de máquina vêm ganhando destaque pelos seus bons resultados. Entretanto, pesquisas mostram que smartphones diferentes registram valores de RSSI diferentes para um mesmo ponto e isso pode prejudicar bastante os resultados obtidos. O objetivo principal deste trabalho é analisar a diferença na acurácia de técnicas de *fingerprinting* (baseadas em três diferentes algoritmos de aprendizagem de máquina) para diferentes modelos de smartphones. Além disso, pretende-se avaliar qual das técnicas possui uma melhor generalização, ou seja, qual técnica apresenta desempenho que menos depende do tipo de smartphone utilizado.

Palavras-chaves: Localização *indoor*; *fingerprinting*; aprendizagem de máquina, comunicações sem fio.

Abstract

Indoor localization techniques have been quite studied in recent years. Although the GPS can give a good approximation of the position of the user in outdoor environments, the degradation of the signal is very large in indoor environments. In this scenario, fingerprinting techniques using RSSI (received signal strength indicator) values have been widely used to locate users because they do not need any additional hardware to do the location. As far as fingerprinting techniques are concerned, machine learning algorithms have been gaining prominence by their good results. However, surveys show that different smartphones record different RSSI values for the same point and this can greatly harm the results obtained. The main objective of this work is to analyze the difference in the accuracy of fingerprinting techniques (based on three different machine learning algorithms) for different models of smartphones. In addition, it is intended to evaluate which of the techniques has a better generalization, i.e., which technique presents the performance that less depends on the type of smartphone used.

Keywords: Indoor location; fingerprinting; machine learning; wireless communications.

Sumário

1 Introdução **6**

1.1 Objetivos 7

1.2 Estrutura do Documento 7

2 Referencial Teórico **8**

2.1 Localização *Indoor* 8

2.1.1 Métodos de localizaçao baseados em hardware 8

2.1.2 Métodos de localização livres de infraestrutura 9

2.2 Aprendizagem de Máquina 11

2.2.1 *k-Nearest Neighbor* 11

2.2.2 *Random Forest* 12

2.2.3 Support Vector Machine 13

2.2.3.1 Classificação de Padrões Linearmente Separáveis 14

2.2.3.2 Classificação de Padrões Não-Linearmente Separáveis 15

2.2.3.3 Support Vector Regression (SVR) 17

3 Metodologia da Pesquisa e Resultados **19**

3.1 Fase 1 - Predição do Prédio 21

3.2 Fase 2 - Predição do Andar 22

3.3 Fase 3 - Predição da Latitude e Longitude 23

3.4 Resultados 23

4 Conclusão **32**

1 Introdução

O interesse por técnicas de localização *indoor* cresceu bastante nos últimos anos. Muitas aplicações precisam ter conhecimento da localização do usuário para prover seus serviços [1]. O GPS é uma boa solução para localização *outdoor*, mas não funciona bem em ambientes *indoor* por causa da atenuação que o sinal sofre devido às estruturas das construções (telhado, paredes e outros objetos) [1]. Por esse motivo, outras soluções foram propostas [2].

Uma delas é a técnica de localização baseada em *fingerprinting*. Este método utiliza valores de intensidade de sinal de rádio frequência (RF) (*received signal strength indicator*, RSSI) para localizar o usuário. Basicamente, a técnica de *fingerprinting* consiste de duas fases: *offline* e *online*. Na fase *offline*, é criada uma base de dados com medições de valores de RSSI referentes a pontos de acesso sem fio existentes. Essa base de dados também inclui coordenadas do local da medição e, eventualmente, outras informações relevantes, tais como número do andar do prédio e modelo do smartphone [3]. Já na fase *online*, as medições do smartphone do usuário são comparadas com as medições armazenadas no banco de dados por meio de alguma métrica de similaridade. As coordenadas geográficas associadas aos valores de RSSI da base de dados mais próximos das medições do smartphone indicarão a posição do usuário.

Uma das maiores vantagens da técnica baseada em *fingerprinting* é que basta a infraestrutura de rede sem fio já existente para sua efetiva utilização, não sendo necessário nenhum hardware adicional [1]. Entretanto, um problema da técnica de *fingerprinting* aplicada à localização *indoor* é que diferentes modelos de smartphones podem detectar não apenas um número diferente de pontos de acesso sem fio, mas também valores de RSSI diferentes para um mesmo ponto de acesso. A maioria das técnicas de *fingerprinting* para localização *indoor* não leva essa questão em consideração [4]. Além disso, a escolha dos algoritmos a serem utilizados e dos seus parâmetros pode afetar a acurácia do método. É difícil encontrar uma técnica que tenha uma boa generalização, de forma a encontrar bons resultados para diferentes tipos de smartphones. Outro problema relevante é que novas técnicas de *fingerprinting* desenvolvidas geralmente utilizam sua própria base de dados [1]. Dessa forma, torna-se difícil realizar uma comparação entre as técnicas já existentes.

1.1 Objetivos

O primeiro objetivo desse trabalho é fazer um estudo comparativo para observar como a acurácia dos algoritmos de aprendizagem de máquina utilizados em técnicas de *fingerprinting* para localização *indoor* muda dependendo do tipo de smartphone utilizado. O segundo objetivo é analisar qual modelo de aprendizagem de máquina apresenta a melhor generalização de acurácia, ou seja, que fornece o melhor desempenho independentemente do tipo de smartphone utilizado.

Para atingir esses objetivos, é necessário utilizar uma base de dados que contenha dados de diferentes tipos de smartphones. Portanto, neste trabalho, é utilizada a base UJIIndoorLoc, disponível no site de acesso público *UCI Machine Learning Repository*[1]. Ela tem mais de 20 mil amostras e foi criada para ser uma base de dados referência na comparação de diferentes metodologias de localização *indoor*[1].

Também é necessário escolher quais os algoritmos de aprendizagem de máquina a serem utilizados. Procurou-se utilizar algoritmos já consolidados para analisar o problema investigado nesse trabalho. Dessa forma, foram escolhidos os algoritmos *k-Nearest Neighbors*, *Random Forest* e *Support Vector Machine*. A fim de se obter os melhores resultados para cada técnica, é feita uma busca dos melhores parâmetros para cada algoritmo.

1.2 Estrutura do Documento

No Capítulo 2 é apresentado o referencial teórico, no qual é descrito o problema mais detalhadamente e os principais trabalhos que propõem soluções relacionadas ao tema. O Capítulo 3 aborda a metodologia do trabalho, bem como os resultados obtidos e as principais análises feitas a partir deles. Por fim, no último Capítulo, apresentam-se as conclusões.

[1]Disponível em https://archive.ics.uci.edu/ml/datasets/ujiindoorloc.

2 Referencial Teórico

Neste Capítulo é feita a contextualização do trabalho. São descritos alguns trabalhos realizados na área de localização *indoor*, bem como a revisão de algumas técnicas de aprendizagem de máquina utilizadas.

2.1 Localização *Indoor*

A localização *indoor* tem se tornado uma área de grande interesse nos últimos anos, principalmente por conta de aplicações para localização de pessoas, intrusos, produtos em supermercados, equipamentos em hospitais, além do uso em robótica, aplicações de propagandas e navegação dentro de edifícios. Muitas aplicações hoje em dia estão embarcadas em smartphones, logo há uma necessidade crescente de se utilizar técnicas para localizar smartphones dentro de ambientes fechados. Além disso, um grande crescimento na utilização de sistemas sem fio tem sido observado nos últimos anos, o que aumenta a demanda por sistemas precisos de localização *indoor* [2]. É desejável que os sistemas de localização *indoor* tenham boa acurácia, tempo de treinamento curto, um bom custo-benefício e sejam robustos [5]. Muitas técnicas de localização se baseiam em parâmetros dos sinais, tais como o tempo de chegada (*Time of Arrival*, ToA), a diferença de tempo de chegada (*Time Difference of Arrival*, TDoA), o nível de intensidade do sinal (RSSI) e o ângulo de chegada (*Angle of Arrival*, AoA) para descobrir a posição do usuário.

Entretanto, quando se trata de localização *indoor*, existem vários desafios a serem enfrentados, são eles: a falta de acurácia do GPS para determinar a localização dentro de edifícios, as atenuações que o sinal sofre por conta de paredes e objetos e a interferência com outros sinais [6]. Para superar esses desafios, várias técnicas foram propostas. Essas técnicas podem ser divididas em duas categorias principais: as baseadas em hardware e as livres de infraestrutura [7].

2.1.1 Métodos de localizaçao baseados em hardware

As técnicas de localização baseadas em hardware utilizam dispositivos adicionais que emitem e recebem sinais. Por conta disso, requerem a implantação de toda a infraestrutura necessária para o seu funcionamento. Podemos citar como exemplos desse tipo de técnica a localização por meio de sensores infravermelhos [8], RFIDs (*Radio-Frequency IDentification*) [9] e dispositivos

ultrassônicos [10]. Geralmente, as técnicas baseadas em hardware tendem a ter uma acurácia maior do que as técnicas que não utilizam dispositivos adicionais. Entretanto, elas geralmente implicam em um alto custo de instalação. Além disso, são difíceis de escalar.

2.1.2 Métodos de localização livres de infraestrutura

Diferente do método descrito anteriormente, as técnicas de localização livres de infraestrutura, como o próprio nome já diz, são construídas com a infraestrutura existente do local, ou seja, não há a necessidade de dispositivos adicionais. Geralmente, as técnicas livres de infraestrutura utilizam as redes Wi-Fi ou *Bluetooth* para sua implementação. Nestas soluções, é preciso que se perceba no ambiente características do local que o distinga de outros locais. Muitos estudos têm sido realizados com base em redes Wi-Fi em virtude da grande disseminação desse tipo de rede sem fio.

As soluções baseadas em redes Wi-Fi podem ser divididas em duas categorias: soluções que constroem modelos matemáticos e soluções que constroem mapas de radiação. A primeira é difícil de se aplicar em situações reais, pois é difícil prever o que acontece com a potência do sinal quando atravessa objetos e paredes [7]. A segunda categoria, também chamada de técnica baseada em *fingerprinting*, é a utilizada neste trabalho. Ela é dividida em duas fases: *online* e *offline*. Na fase *offline*, ocorre a coleta dos dados característicos do local, formando assim uma base de dados. Na fase *online* é realizada a comparação entre os dados da base e o dado do usuário para descobrir sua localização.

Um ponto negativo do *fingerprinting* é que o processo de coleta dos dados na fase *offline* é trabalhoso e consome muito tempo. Além disso, o principal desafio desta técnica é a sensibilidade à mudanças no ambiente. Mudanças na posição dos móveis dentro do prédio, difração e reflexão resultam em mudanças na propagação do sinal [11].

Há dois trabalhos bastante conhecidos que utilizam a técnica do *fingerprinting*: o RADAR [12] e o HORUS [13]. O RADAR foi o primeiro trabalho a propor o uso da potência de sinal para identificar localização. Após o RADAR, muitos outros trabalhos seguiram o mesmo caminho, incluindo o trabalho aqui proposto. O RADAR combina medições empíricas com modelos de propagação de sinal para determinar a localização do usuário. Assim como todas as técnicas de *fingerprinting*, ele possui as fases *online* e *offline*. Na fase *offline*, é feita a coleta dos *fingerprints* no edifício. Os *fingerprints* relacionam a potência do sinal Wi-Fi com a localização. Assim, para cada localização, são conhecidas as potências de sinal que cada roteador emite. Quanto

maior o número de *fingerprints*, melhores são os resultados da técnica. Na fase *online* o usuário requisita a localização e é feita uma comparação entre os *fingerprints* obtidos na fase anterior com a medição atual. Essa comparação é feita através do cálculo da distância euclidiana. Descobriu-se nesse trabalho alguns fatos interessantes como, por exemplo, que a força do sinal varia dependendo da orientação do usuário e da direção em que ele está seguindo.

Assim como no RADAR, o HORUS usa a potência de sinal Wi-Fi no seu sistema de localização *indoor*. O HORUS modela as distribuições de força do sinal recebidas dos pontos de acesso utilizando distribuições paramétricas e não paramétricas. Portanto, ele traz uma abordagem mais probabilística em relação ao RADAR. Dado um conjunto X de pontos que representam as posições do *fingerprints*, o HORUS tenta encontrar um ponto x pertencente a X que maximiza a probabilidade do usuário estar naquele local. Além disso, o HORUS tenta diminuir os requisitos computacionais dos algoritmos utilizados. No trabalho também foi apresentada uma comparação entre o RADAR e o HORUS e, para uma mesma base de dados, o último obteve melhores resultados.

Embora modelos de propagação sejam amplamente utilizados para resolver problemas de localização, geralmente seus resultados não são tão bons. Para resolver esse problema, recentemente tem se usado diversas técnicas de aprendizagem de máquina na modelagem de problemas de localização. As máquinas de aprendizagem aprendem a partir de exemplos e fazem predições a partir deles. O trabalho em [14] propõe uma SVR (*Support Vector Regression*) multidimensional para resolver o problema de localização. O RADAR introduziu o k-NN (*k-Nearest Neighbor*) para fazer localização em ambientes *indoor*. Já [15] propõe um sistema de localização que utiliza RFID para encontrar pacientes dentro de um hospital. Para tal, os autores utilizam uma *Random Forest*. Algumas outras técnicas muito usadas em sistemas de localização são: *Multilayer Perceptron* (MLP), *Naive Bayes* e *AdaBoost*.

Para se comparar os resultados obtidos de diferentes máquinas de aprendizagem, é necessário executá-las sobre uma mesma base de dados. Alguns sites, como *UCI Machine Learning Repository*, oferecem bases de dados públicas com esse objetivo. Na área de localização *indoor*, entretanto, não existem muitas bases públicas. Portanto, neste trabalho, foi utilizada a base de dados UJIIndoorLoc [1]. Ela foi criada com dois propósitos principais: resolver o problema descrito anteriormente e ser uma referência para comparar diferentes metodologias de localização *indoor*. A UJIIndoorLoc foi a primeira base de dados pública nessa área, construída em 2014, e se encontra disponível no repositório da *UCI Machine Learning*.

Apesar das técnicas de *fingerprinting* estarem mostrando resultados satisfatórios, elas atualmente assumem que os dispositivos celulares vão ter a mesma performance, não importando qual modelo seja usado. Isso não é verdade. [4] mostrou que há diferenças significativas entre as medições de RSSI para cada celular. Até mesmo dispositivos do mesmo fornecedor não têm medições semelhantes. Mostrou-se também que alguns dispositivos não têm um padrão nas medições tornando, dessa forma, esses celulares indesejáveis para os sistemas que utilizam *fingerprinting*. Outra contribuição desse mesmo trabalho foi a descoberta de que celulares que apresentam banda de sinal de 5GHz identificam melhor o nível de sinal e mantêm um padrão melhor nas medições em relação a dispositivos que possuem banda de 2.4GHz.

2.2 Aprendizagem de Máquina

Nesta Seção serão apresentados os algoritmos de aprendizagem de máquina utilizados no trabalho proposto. Todos os algoritmos mostrados podem ser usados tanto para problemas de classificação, quanto para de regressão. Na classificação a saída da máquina de aprendizado é uma variável discreta que representa alguma das *labels* das classes. Enquanto isso, na regressão, a saída é um valor contínuo.

2.2.1 *k-Nearest Neighbor*

O *k-Nearest Neighbor* (k-Vizinhos Mais Próximos) é um método baseado em instâncias. A aprendizagem nos algoritmos baseados em instâncias consiste somente em armazenar os dados de treinamento. Quando uma instância precisa ser testada, o conjunto de treinamento é recuperado da memória para a classificação ser realizada. Uma desvantagem nesses tipos de algoritmos é que o custo computacional deles pode ser muito alto dependendo da quantidade de dados de treinamento. Outro ponto negativo é que todos os atributos são considerados na classificação, em vez de se considerar somente os mais importantes [16]. Dentre os algoritmos baseados em instâncias, o k-NN é um dos mais simples e um dos mais conhecidos em reconhecimento de padrões para classificação não paramétrica supervisionada [17]. É considerado um *lazy learner* porque ele espera por uma consulta para generalizar o modelo.

Para classificação, o algoritmo funciona da seguinte maneira: quando uma instância de teste chega, o algoritmo procura os k vizinhos mais próximos dele. A classe mais frequente entre os k vizinhos será a classe da instância de teste, onde k é uma constante definida pelo usuário. A proximidade entre a instância de teste e as outras instâncias é calculada através de uma função

de distância. Uma das mais comumente utilizadas é a distância euclidiana. O Algoritmo 1 descreve com mais detalhes o funcionamento do k-NN para classificação.

Algorithm 1 k-Nearest Neighbor - Classificação.

Algoritmo de treinamento:
    Para cada exemplo de treino $(x, f(x))$, adicione o exemplo na lista *exemplos_treino*.
Algoritmo de teste:
    Dada uma instância de teste x_t a ser classificada:
        Sejam $x_1..x_k$ as k instâncias de *exemplos_treino* mais próximos de x_t.
        Retorne:
            $$\hat{f}(x_t) \leftarrow \underset{v \in V}{argmax} \sum_{i=1}^{k} \delta(v, f(x_i))$$
    onde $\delta(a, b) = 1$ se $a = b$ e $\delta(a, b) = 0$ caso contrário.

Para a regressão, ao invés de procurar qual a classe mais frequente, o algoritmo faz uma média dos valores dos k vizinhos mais próximos. Esse será o valor atribuído à instância de teste. A proximidade entre as instâncias é calculada da mesma forma que na classificação. O Algoritmo 2 mostra o funcionamento do k-NN para regressão.

Algorithm 2 k-Nearest Neighbor - Regressão.

Algoritmo de treinamento:
    Para cada exemplo de treino $(x, f(x))$, adicione o exemplo na lista *exemplos_treino*.
Algoritmo de teste:
    Dada uma instância de teste x_t a ser classificada:
        Sejam $x_1..x_k$ as k instâncias de *exemplos_treino* mais próximos de x_t.
        Retorne:
            $$\hat{f}(x_t) \leftarrow \sum_{i=1}^{k} \frac{f(x_i)}{k}$$

Existem vários refinamentos para se obter melhores resultados com o k-NN. Um deles é ponderar a contribuição de cada um dos k vizinhos de acordo com sua distância. Por exemplo, o peso pode ser o inverso do quadrado da distância entre a instância de testes e o vizinho. Dessa forma, os vizinhos mais próximos terão um efeito maior na resposta do que os mais afastados [16].

2.2.2 *Random Forest*

O *Random Forest* é uma melhoria da técnica *Bagging*. Também chamado de *bootstrap aggregation*, o *Bagging* tenta reduzir a variância de uma função de predição [18]. Essa técnica é utilizada principalmente para diminuir a variância em Árvores de Decisão. Para isso, o algoritmo seleciona B vezes os exemplos do conjunto de treinamento e treina B árvores com esses exemplos. Essa seleção é feita de forma aleatória e com substituição. A predição final é obtida

através da média das predições de todas as árvores no caso da regressão ou tomando a classe majoritária no caso da classificação. O *Bagging* tem obtido ótimos resultados combinando centenas e até milhares de árvores [19].

Random Forests se diferem do *Bagging* em somente um aspecto: o algoritmo da Árvore de Decisão é modificado para selecionar a cada divisão de nó somente alguns atributos escolhidos aleatoriamente e não todos. O objetivo disso é evitar que as árvores sejam correlacionadas. Se um dos atributos é muito forte, então ele provavelmente será escolhido por várias árvores. Consequentemente, as árvores vão ser muito similares entre si, fazendo elas se tornarem correlacionadas [19]. O Algoritmo 3 mostra o pseudocódigo para o *Random Forest* [18].

Se os dados têm p atributos, é indicado usar para classificação $\lfloor\sqrt{p}\rfloor$ atributos na divisão. Para a regressão, esse valor muda para $\lfloor p/3\rfloor$ atributos [18]. *Random Forests* também podem ser utilizadas para medir a importância de cada variável de um problema. Elas são robustas a ruídos e podem ser aplicadas em problemas não supervisionados.

Algorithm 3 *Random Forest* para Classificação e Regressão.

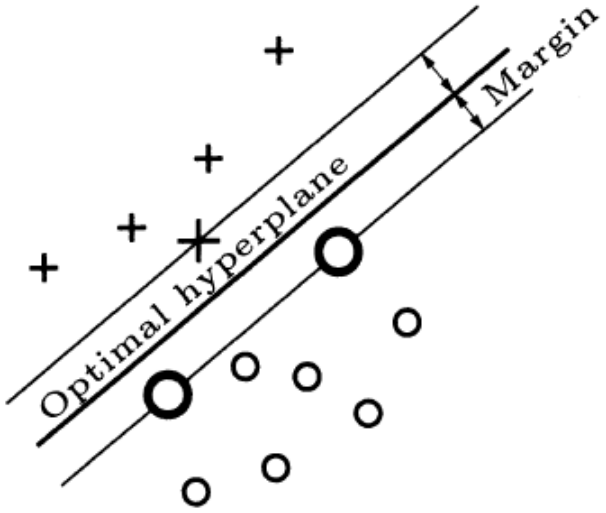
1: **For** $b = 1$ **to** B:
2: (a) Faça uma seleção aleatória dos dados de treinamento.
3: (b) Crie uma árvore de decisão T_b com os dados selecionados, recursivamente repetindo os seguintes passos para cada nó terminal da árvore, até o nó de tamanho mínimo ser encontrado.
4: (i) Selecione m atributos aleatóriamente dos p atributos da base de treinamento.
5: (ii) Selecione o melhor atributo dentre os m.
6: (iii) Divida o nó em dois nós filhos.
7: A saída será o conjunto de árvores $\{T_b\}_1^B$.
8: Para fazer a predição de um novo ponto x:
9: Para Regressão: $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.
10: Para Classificação: Seja $\hat{C}_b(x)$ a classe predita da *bth* árvore da *Radom Forest*. Então $\hat{C^B}(x)$ = votos majoritários $\{\hat{C}_b(x)\}_1^B$.

2.2.3 Support Vector Machine

As Máquinas de Vetores de Suporte (SVM) vêm recebendo cada vez mais atenção na área de Aprendizagem de Máquina. A técnica foi introduzida por Vapnik [20], em 1995. Ela pode resolver tanto problemas linearmente separáveis como não-linearmente separáveis. Além disso pode ser usada tanto para regressão quanto para classificação. O objetivo da SVM é encontrar um hiperplano que separe o máximo possível dados de classes diferentes. Esse hiperplano é chamado de hiperplano ótimo. A distância entre o hiperplano até a primeira instância de

cada classe é chamada de margem. A margem determina o quão bem duas classes podem ser separadas. Na Figura 1 é mostrado o hiperplano ótimo e suas margens. As instâncias que se encontram em cima da reta que delimita a margem são chamadas de vetores de suporte.

Figura 1: O hiperplano ótimo é o que separa os dados de classes diferentes com a maior margem possível.



Fonte: The Nature of Statistical Learning Theory, Vapnik, 1995, p.128

2.2.3.1 Classificação de Padrões Linearmente Separáveis

A classificação linear é implementada pelo uso de uma função real $f : X \subseteq R^N \rightarrow R^N$. Se, dada uma entrada x, e $f(x) \geqslant 0$, então é atribuída uma classe positiva a x. Caso contrário, é atribuída uma classe negativa. Uma função linear $f(x)$ pode ser representada pela equação 1.

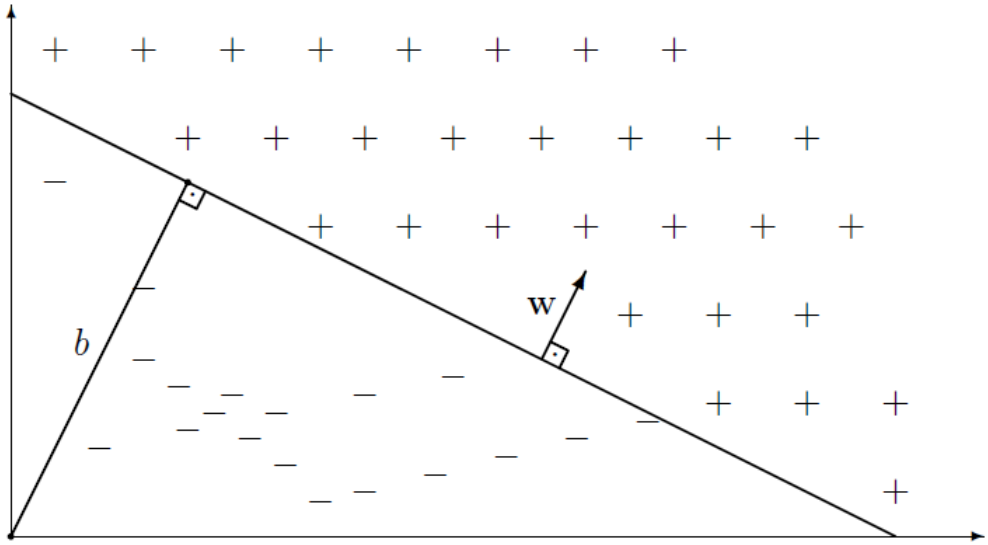
$$f(x) = \langle \mathbf{w}.\mathbf{x} \rangle + b \tag{1}$$

Onde $\mathbf{w}$ é o vetor peso e b é o *bias*. O vetor $\mathbf{w}$ tem direção perpendicular ao hiperplano, enquanto que a variação do valor b move o hiperplano paralelamente a ele mesmo, como pode ser visto na Figura 2. Ambos, $\mathbf{w}$ e b, são definidos no processo de aprendizagem [21].

O hiperplano ótimo é definido pela seguinte equação:

$$f(x) = \langle \mathbf{w}.\mathbf{x} \rangle + b = 0 \tag{2}$$

Figura 2: **w** e b em um hiperplano que divide duas classes.



Fonte: Máquinas de Vetores Suporte na Classiflcação de Impressões Digitais, Lima, 2002, p.48

A separação das classes pode ser representada da seguinte forma:

$$\langle \mathbf{x}_i.\mathbf{w} \rangle + b \geq +1, \text{para } y_i = +1 \tag{3}$$

$$\langle \mathbf{x}_i.\mathbf{w} \rangle + b \leq -1, \text{para } y_i = -1 \tag{4}$$

Seja d_+ e d_- a distância dos vetores de suporte positivos e negativos, respectivamente, até o hiperplano. A margem pode ser representada pela equação 5 [22].

$$\rho = (d_+ + d_-) = \frac{2}{\|\mathbf{w}\|} \tag{5}$$

O hiperplano é, então, obtido pela minimização da norma $\|\mathbf{w}\|$. Este problema de otimização pode ser resolvido por multiplicadores de Lagrange [22]. Realizando os cáculos necessários, detalhados em [22], chegamos a equação utilizada para classificar um novo padrão **z**:

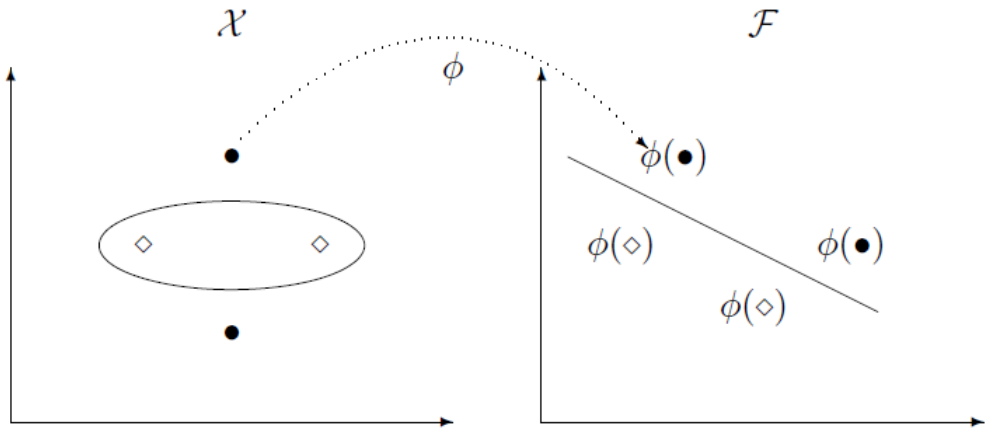
$$sinal(\langle \mathbf{w}.\mathbf{z} \rangle + b) \tag{6}$$

2.2.3.2 Classificação de Padrões Não-Linearmente Separáveis

A maioria dos problemas reais não apresenta linearidade. Para trabalhar com esse tipo de problema, a SVM mapeia o conjunto de treinamento em um espaço em que os dados possam ser linearmente separáveis. Esse mapeamento é feito por meio das funções de *kernel*.

O espaço de características é o espaço de mais alta dimensionalidade onde serão mapeados o conjunto de entrada por meio de uma função Φ para, assim, obter um novo conjunto de treinamento que pode ser linearmente separável. Um exemplo dessa idea é representada na Figura 3. Onde um conjunto não-linearmente separável X é transformado em um conjunto linearmente separável F.

Figura 3: Mapeamento do conjunto de treinamento X para um conjunto de treinamento F. Os dados não podem ser linearmente separados em X mas podem em F.



Fonte: Máquinas de Vetores Suporte na Classiflcação de Impressões Digitais, Lima, 2002, p.57

Para classificar um novo padrão $\mathbf{z}$, nesse caso, é utilizada a seguinte equação:

$$sinal(\langle \mathbf{w}.\mathbf{\Phi}(\boldsymbol{z})\rangle + b) \tag{7}$$

Para evitar cálculos computacionais em um espaço dimensional muito grande, são utilizadas funções de *kernel*. Essas funções calculam o produto interno de dois dados de entrada x_i e x_j no espaço de características, como pode ser visto na equação 8.

$$K(x_i, x_j) = \langle \Phi(x_i).\Phi(x_j)\rangle \tag{8}$$

Para uma função ser uma função de *kernel* ela deve obedecer as condições estabelecidas pelo Teorema de Mercer [20]. O Teorema diz que uma função é de *kernel* se a matriz K é positivamente definida. Onde K é obtida por:

$$K = K_{ij} = K(x_i, x_j) \tag{9}$$

Uma matriz é positivamente definida se seus autovalores são maiores que zero.

Os *kernels* utilizados neste trabalho foram *sigmoid*, *linear*, *polinomial* e *RBF (Radial Basis Function)*. Suas funções são mostradas, juntamente com seus parâmetros, na Tabela 1.

Tipo de *kernel*	Função $K(x_i, x_j)$	Parâmetros
Linear	$\langle x_i.x_j \rangle$	não possui
Sigmoid	$tanh(\delta\langle x_i.x_j\rangle + \kappa)$	δ e κ
Polinomial	$(\delta\langle x_i.x_j\rangle + \kappa)^d$	δ, κ e d
RBF	$exp(-\delta\|x_i - x_j\|)^2$	δ

Tabela 1: Funções de *kernel* usadas neste trabalho.

2.2.3.3 Support Vector Regression (SVR)

Uma versão da SVM para regressão foi proposta em 1996, no trabalho [23]. O objetivo da SVR é construir um hiperplano que passe o mais próximo possível dos dados [24]. Ou seja, achar uma função $f(x)$ que desvie de no máximo um valor ϵ do valor alvo [25]. Portanto, o hiperplano deve ter uma norma pequena e deve minimizar a soma das distâncias dos dados a ele [6].

Considere l pares de dados de treinamento $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_l}, y_l)$, onde $\boldsymbol{x_i}$ é um vetor n-dimensional e y_i é um valor contínuo. Esse problema pode ser descrito como um problema de otimização, onde se tenta minimizar $\frac{1}{2}\|w\|^2$ e as restrições são:

$$y_i - \langle w, x_i\rangle - b \leq \epsilon \tag{10}$$

$$\langle w, x_i\rangle + b - y_i \leq \epsilon \tag{11}$$

Essa restrição assume que uma função f existe tal que aproxima todos os pares $(\boldsymbol{x_i}, y_i)$ com precisão ϵ. No entanto, nem sempre esse é o caso. Além disso, às vezes, é desejável permitir alguns erros. Portanto, pode-se introduzir variáveis de folga ξ_i e ξ_{i^*} que ajudam a lidar com pontos a uma distância maior que ϵ. A partir disso, chegamos a seguinte formulação [25]:

$$\text{Minimizar: } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{12}$$

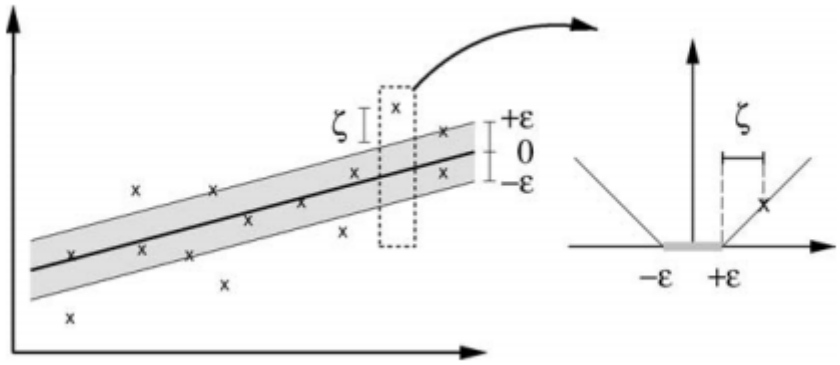
$$\text{Com a restrição: } \begin{cases} y_i - \langle w, x_i\rangle - b \leq \epsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{13}$$

Onda a constante C determina um ponto de equilíbrio entre a maximização da margem e a minimização do erro de classificação.

A formulação anterior corresponde a lidar com uma função chamada ϵ-*insensitive* [25], representada na Figura 4 e descrita por:

$$|\xi|_\epsilon = \begin{cases} 0, \text{ se } |\xi| \leq \epsilon \\ |\xi| - \epsilon, \text{ caso contrário} \end{cases} \tag{14}$$

Figura 4: Função de perda ϵ-*insensitive*.



Fonte: *A tutorial on support vector regression*, Smola e Scholkopf, 2003, p.2

O problema pode então ser resolvido por multiplicadores de Lagrange, como mostrado em [26]. No caso de regressão linear, o hiperplano obtido é dado por:

$$h(\boldsymbol{x_i}, \boldsymbol{w}) = \sum_{j=1}^{l} (\boldsymbol{\alpha} - \boldsymbol{\alpha^*}) \langle \boldsymbol{x_j}, \boldsymbol{x_i} \rangle + b \tag{15}$$

No caso da regressão não-linear, o hiperplano é dado por::

$$h(\boldsymbol{x_i}, \boldsymbol{w}) = \sum_{j=1}^{l} (\boldsymbol{\alpha} - \boldsymbol{\alpha^*}) \langle \Phi(\boldsymbol{x_j}), \Phi(\boldsymbol{x_i}) \rangle + b \tag{16}$$

Onde: $\boldsymbol{\alpha}$ e $\boldsymbol{\alpha^*}$ são os multiplicadores de Lagrange.

Assim como no caso da classificação, as funções de *kernel* também são utilizadas para mapear o conjunto de treinamento no espaço de características.

3 Metodologia da Pesquisa e Resultados

Neste Capítulo, é apresentada a metodologia utilizada para a execução dos experimentos e os resultados obtidos. Inicialmente, é descrita a base de dados utilizada e o pré-processamento que foi realizado nela. Em seguida, descreve-se a implementação dos algoritmos de aprendizagem de máquina e o procedimento empregado para localizar o usuário. Por fim, são apresentados os resultados.

A base de dados utilizada no trabalho foi a UJIIndoorLoc. Trata-se de uma base de dados pública de localização *indoor*, que possui 21.049 amostras, sendo 19.938 para treinamento e 1.111 para validação e teste. Neste trabalho, foi utilizado somente o conjunto de treinamento da base, pois o conjunto de validação e teste não possui a posição dos celulares. Logo, não seria possível fazer nenhuma avaliação sobre a acurácia dos algoritmos utilizados. Os dados da base UJIIndoorLoc foram coletados por mais de 20 usuários utilizando 25 modelos de celulares *Android* diferentes. As coletas foram feitas em três prédios da *Universitat Jaume I*, cobrindo uma área de 108.303 m^2 e utilizando 4 ou 5 andares, dependendo do prédio. A base de dados foi construída medindo-se os níveis de RSSI com uma aplicação *Android*. A base de dados contém os níveis de RSSI dos pontos de acesso sem fio (WAPs - *Wireless Access Point*), as coordenadas geográficas das amostras (latitude, longitude e andar), o identificador do prédio e o identificador da sala em que foram coletadas as amostras, um identificador para saber se o usuário encontra-se no corredor ou na sala, o identificador do smartphone e o horário em que foi feita a medição. Na base há um total de 520 WAPs diferentes, porém nem todos são detectados na mesma medição. O número médio de WAPs capturados por medição é em torno de 18. A latitude e longitude foram medidas em metros e utilizou-se a norma WGS 84 (*World Geodetic System*). Cada ponto de referência foi medido por pelo menos dois usuários [1]. A base UJIIndoorLoc foi escolhida para este trabalho pelo fato de possuir um grande número de amostras de vários modelos de smartphones diferentes.

Na construção da base de dados, cada smartphone coletou quantidades diferentes de amostras. Essa quantidade variou de 192 até 4835 amostras. Em vista disso, somente os 4 celulares que tinham o maior número de medições foram escolhidos para o estudo. Essa decisão baseou-se no fato de que uma menor quantidade de amostras comprometeria a acurácia das máquinas de aprendizagem. Dito isso, os smartphones escolhidos foram os modelos LT22i, com 4835 amostras, HTC Wildfire S, com 4516 amostras, GT-S6500, com 1596 amostras e GT-S5360,

com 1383 amostras. Os modelos são mostrados na Figura 5. Para tornar a comparação entre os smartphones escolhidos mais justa, foi realizada uma subamostragem nos conjuntos dos celulares que tinham o maior número de amostras para, desse modo, se reduzirem ao tamanho do conjunto do modelo GT-S5360, que continha a menor quantidade. A retirada de amostras foi feita de forma aleatória.

Figura 5: Modelos utilizados no trabalho.

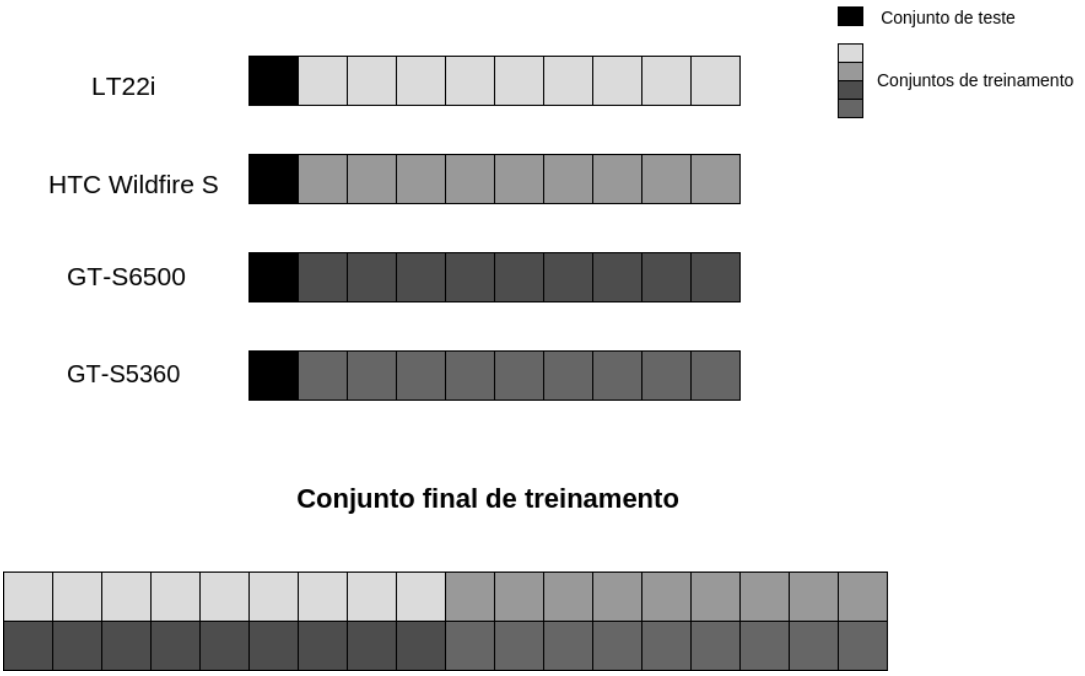
Fonte: Elaborado pela autora.

Na realização das comparações entre os celulares, foi necessário tomar algumas decisões importantes, principalmente em relação a divisão do conjunto de dados e aos parâmetros utilizados por cada máquina de aprendizagem. Neste Capítulo serão descritas cada uma delas.

Com o intuito de avaliar a capacidade de generalização dos modelos, foi utilizada a técnica de validação cruzada *K-fold*, na qual o conjunto total de dados é dividido em K subconjuntos do mesmo tamanho e, a partir disso, um subconjunto é usado para testes e o restante para treinamento. Esse processo é repetido K vezes e, em cada uma das iterações, os subconjuntos de teste e treino são alterados. A Figura 6 ilustra como a divisão do conjunto em treinamento e teste para um K específico é feita neste trabalho. A divisão em *K-folds* é feita para cada conjunto de celular. Contudo, após a divisão, os subconjuntos de treinamento de todos os celulares são dispostos em um único conjunto. Os subconjuntos de teste continuam separados para cada celular. Para o trabalho proposto, o valor de K utilizado é 10.

Objetivando encontrar o usuário, o algoritmo implementado passa por três fases principais: na primeira, busca-se em qual dos três prédios o usuário se encontra; na segunda, deseja-se saber em qual andar do prédio identificado na primeira fase o usuário está e, por fim, na terceira fase, as coordenadas (latitude e longitude) do usuário são obtidas. As duas primeiras fases consistem em uma classificação, enquanto na terceira são feitas regressões. As três fases utilizam todo o conjunto de treinamento selecionado no *K-fold*. A metodologia explicada acima é ilustrada na

Figura 6: Nove *folds* de cada celular são separados para treino e um para teste. Após isso, todos os *folds* de treinamento são misturados em um único conjunto. Os *folds* de teste continuam separados.



Fonte: Elaborado pela autora.

Figura 7.

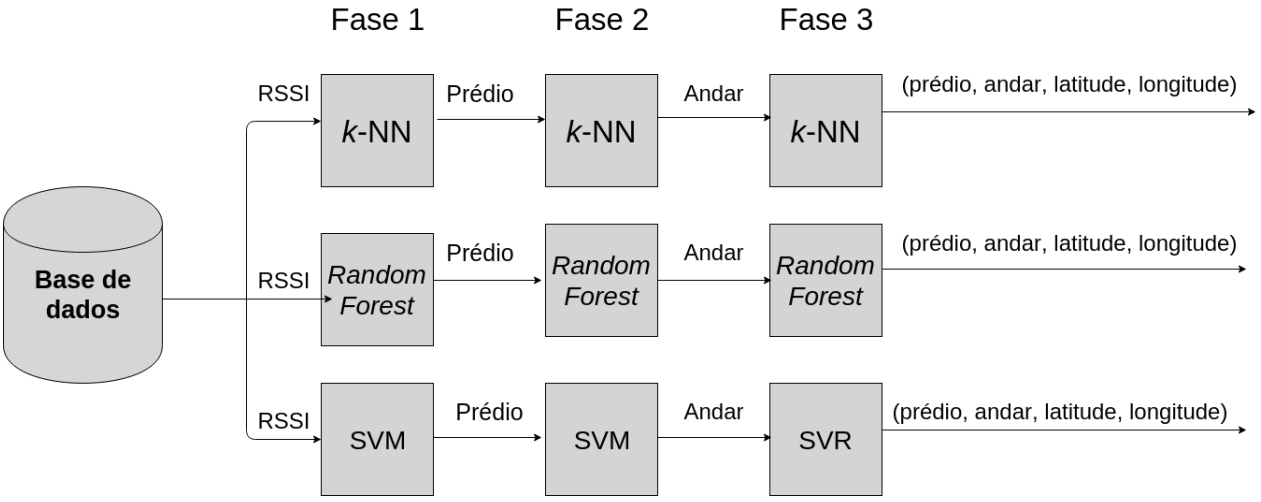
Com o objetivo de criar modelos para prever a localização do usuário e compará-los, em cada uma das fases foram utilizados três algoritmos de aprendizagem de máquina bastante conhecidos em problemas de *fingerprinting*: o k-NN, o *Random Forest* e a SVM. Os algoritmos foram implementados em *python*, na versão 2.7.9 e utilizando a biblioteca *scikit-learn*[2].

3.1 Fase 1 - Predição do Prédio

Na fase 1, são usados classificadores que têm como entrada os valores de RSSI para os WAPs identificados pelo smartphone e têm como saída o identificador do prédio em que o usuário se encontra. Portanto, nessa fase, considera-se que o algoritmo acerta se a previsão do identificador do prédio estiver correta. Para o algoritmo k-NN, foram testados diferentes valores de k a fim de se encontrar o melhor resultado. Os valores testados foram 1,3,5,7 e 10, e o melhor parâmetro encontrado foi $k = 1$. Para medir a similaridade, foi utilizada a distância euclidiana entre os pontos e calculada a função objetivo de forma ponderada pelo inverso da distância. Com a função ponderada, podemos fazer com que vizinhos mais próximos tenham

[2]Disponível em http://scikit-learn.org/stable/.

Figura 7: A figura mostra as três etapas do processo até chegar à predição da localização do usuário. Primeiro é encontrado o prédio que o usuário está (Fase 1), depois o andar (Fase 2) e em seguida as coordenadas (Fase 3).



Fonte: Elaborado pela autora.

um peso mais forte do que os mais distantes.

O modelo de *Random Forest* utilizado nessa fase foi implementado com 50 árvores. Quanto maior o número de árvores, mais o resultado tende a melhorar. Empiricamente, descobriu-se que a quantidade de árvores igual a 50 era um bom balanceamento entre acurácia e tempo computacional. Todos os outros parâmetros do modelo permaneceram com o valor padrão da biblioteca *scikit-learn* para o *Random Forest*[3].

Na SVM, foi feita uma busca exaustiva a fim de encontrar os parâmetros que levassem aos melhores resultados. Os parâmetros variados foram: o *kernel*, o parâmetro de penalidade C e o *gamma*. Os outros parâmetros ficaram com os valores padrões da biblioteca[4]. Os *kernels* testados foram *sigmoid*, *linear*, *polinomial* e *RBF*. O C variou em potências de 2, desde o valor de 2^{-3} até o valor de 2^{8}. O *gamma* também variou em potências de 2, os valores foram de 2^{-5} até 2^{3}. Os melhores parâmetros para essa fase foram: *kernel sigmoid*, $C = 256$ e $gamma = 4$.

3.2 Fase 2 - Predição do Andar

Na segunda fase, a entrada dos classificadores continua a mesma, mas a saída agora é o identificador do andar em que o usuário se encontra. Entretanto, uma predição só é considerada correta se ambos identificador do prédio e do andar estiverem corretos, portanto essa fase

[3]http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[4]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

depende da anterior. Os classificadores usados continuam sendo o k-NN, o *Random Forest* e a SVM. Os parâmetros usados nas funções do k-NN e do *Random Forest* também são iguais aos da fase 1. O melhor k para o k-NN nessa fase foi 1, assim como na primeira fase. Os parâmetros da SVM mudam pois, novamente, realizamos a busca exaustiva, da mesma forma e com as mesmas configurações em que foi realizada anteriormente. Para essa fase os melhores parâmetros encontrados foram: *kernel RBF*, $C = 16$ e $gamma = 2$.

3.3 Fase 3 - Predição da Latitude e Longitude

A terceira fase consiste em achar a latitude e longitude por meio da regressão dos três métodos: k-NN, *Random Forest* e SVR, que é a versão para regressão da SVM. A entrada das máquinas de aprendizagem é igual a das fases anteriores, porém aqui são utilizados dois regressores para se obter a saída em cada algoritmo: um para prever a latitude e outro para prever a longitude. Os k-NNs usados para a regressão também tiveram a quantidade de vizinhos variada como na fase 1. O melhor parâmetro encontrado para ambos os regressores foi $k = 5$. As outras configurações se mantiveram iguais às da fase 1. Já o *Random Forest* nesta fase utiliza 500 árvores para ambos os regressores em vez de 50, pois os resultados com um número menor de árvores não estavam sendo muito satisfatórios. Os outros parâmetros se mantiveram iguais aos da fase 1.

Novamente, houve uma busca exaustiva para achar os melhores parâmetros para a SVR, da mesma forma como foi feita nas outras duas fases. Os melhores parâmetros obtidos para o regressor da longitude foram: *kernel RBF*, $C = 256$ e $gamma = 8$. Para o regressor da latitude foram: *kernel RBF*, $C = 64$ e $gamma = 8$.

Para calcular o erro entre as coordenadas preditas e as reais, foi feito o cálculo da distância geográfica entre as duas localizações. A distância foi calculada com as fórmulas de Vincenty, que são dois métodos iterativos usados para calcular a distância entre dois pontos na superfície de um esferoide [27].

3.4 Resultados

Como já mencionado no Capítulo 1, os objetivos deste trabalho são comparar a capacidade de generalização dos algoritmos para diferentes smartphones e comparar para uma mesma técnica quais deles apresentam os melhores resultados. Para atingir esses propósitos, a performance dos celulares para as três máquinas de aprendizagem foi avaliada a partir de simulações com-

putacionais. Portanto, nesta Seção serão apresentados os principais resultados obtidos, bem como análises comparativas feitas a partir deles. Primeiramente serão comparadas as técnicas que obtiveram melhores resultados para cada smartphone e depois comparados os smartphones que obtiveram melhores resultados para cada técnica.

Assim, como descrito anteriormente, três fases foram utilizadas para encontrar a posição do usuário. Portanto, a metodologia proposta neste trabalho tem como resultado final uma tupla que representa a localização da seguinte forma: (b, f, x, y), onde b é o identificador do prédio, f é o identificador do andar, x é a latitude e y é a longitude predita.

Na fase 1, a métrica utilizada foi a taxa de acerto, ou seja, a quantidade de amostras classificadas com o identificador correto do prédio dividido pela quantidade de amostras de teste totais. Na fase 2, a métrica utilizada também foi a taxa de acerto, mas com uma pequena diferença: uma amostra nessa fase é considerada correta se, além de ter o andar classificado corretamente, o prédio daquela amostra também tiver sido classificado corretamente na fase anterior. A taxa de acerto final de cada uma dessas fases foi a média da taxa de acerto das 10 iterações do *K-fold*.

Para os resultados da fase 3, foi utilizado o erro médio absoluto como métrica de desempenho. O erro foi calculado como a distância (em metros) entre as coordenadas preditas e as reais. Para achar a distância entre esses pontos foi utilizada a função de Vincenty [27].

As Tabelas 2, 3 e 4 apresentam os resultados para cada fase. Elas mostram os resultados para cada combinação de smartphone e técnica utilizada no trabalho. O melhor resultado para cada smartphone foi colocado em negrito. Os dados nas Tabelas são apresentados como a média das 10 iterações do *K-fold* e desvio padrão.

A Tabela 2 mostra a taxa de acertos para a fase 1. Pode-se notar que em todos os smartphones testados, utilizando qualquer uma das técnicas, a taxa de acerto é maior que 99%. Entretanto, a SVM obteve resultados um pouco abaixo em relação às outras duas técnicas.

Smartphone\Método	k-NN	*Random Forest*	SVM
GT-S5360	$\mathbf{1,0 \pm 0,0}$	$0,9993 \pm 0,0022$	$0,9978 \pm 0,0046$
GT-S6500	$\mathbf{1,0 \pm 0,0}$	$\mathbf{1,0 \pm 0,0}$	$0,9993 \pm 0,0022$
HTC Wildfire S	$0,9993 \pm 0,0022$	$\mathbf{1,0 \pm 0,0}$	$0,9971 \pm 0,0035$
LT22i	$\mathbf{1,0 \pm 0,0}$	$\mathbf{1,0 \pm 0,0}$	$0,9971 \pm 0,0058$

Tabela 2: Taxa de acerto média, com desvio padrão - Fase 1.

Na Tabela 3 são apresentadas as taxas de acerto para a fase 2, ou seja, levando em consideração o acerto do prédio e do andar. Este fato justifica a taxa de acertos nesta fase não

ser tão boa quanto na anterior, embora os resultados ainda continuem satisfatórios. É possível perceber que, para todos os smartphones, o *Random Forest* é o algoritmo que obteve a melhor acurácia. Além da média ser menor, a dispersão em torno da média também é menor do que nos outros algoritmos, ou seja, os dados tendem a estar mais próximos da média. Na maioria dos smartphones, a técnica SVM obteve resultados melhores que o k-NN, o que divergiu da fase anterior, onde ela teve uma acurácia um pouco pior que as demais técnicas. Isso significa que a classificação da SVM somente para o andar, sem levar em consideração o prédio, tem uma taxa de acerto maior que a do k-NN.

Smartphone\Método	k-NN	*Random Forest*	SVM
GT-S5360	$0,9870 \pm 0,0116$	$\mathbf{0,9971 \pm 0,0035}$	$0,9899 \pm 0,0081$
GT-S6500	$0,9949 \pm 0,0056$	$\mathbf{1,0 \pm 0,0}$	$0,9884 \pm 0,0058$
HTC Wildfire S	$0,8966 \pm 0,0313$	$\mathbf{0,9855 \pm 0,0097}$	$0,9393 \pm 0,0074$
LT22i	$0,9631 \pm 0,0164$	$\mathbf{0,9819 \pm 0,0163}$	$0,9653 \pm 0,0191$

Tabela 3: Taxa de acerto média, com desvio padrão - Fase 2.

A Tabela 4 apresenta os resultados da terceira fase. Nesta fase, os resultados para cada combinação de técnica-smartphone já diferem mais do que nas outras. Os melhores resultados variaram entre as técnicas *Random Forest* e k-NN. Para os modelos GT-S5360 e HTC Wildfire S, a *Random Forest* foi a técnica que obteve melhor acurácia. Já para os modelos GT-S6500 e LT22i, a acurácia do k-NN foi melhor em relação aos outros métodos. A SVM obteve os piores resultados para todos os smartphones, com o maior erro ultrapassando 10 metros. Já o menor erro entre todas as combinações foi de 1,6729 metros para o smartphone GT-S6500, utilizando o k-NN.

Smartphone\Método	k-NN	*Random Forest*	SVM
GT-S5360	$5,8646 \pm 0,5057$	$\mathbf{5,0692 \pm 0,3365}$	$8,1017 \pm 0,4536$
GT-S6500	$\mathbf{1,6729 \pm 0,2553}$	$2,1824 \pm 0,4640$	$3,5614 \pm 0,2869$
HTC Wildfire S	$8,3593 \pm 0,9261$	$\mathbf{8,3527 \pm 0,8874}$	$10,393 \pm 1,1246$
LT22i	$\mathbf{6,2263 \pm 1,1690}$	$6,4839 \pm 0,8615$	$6,8986 \pm 0,8225$

Tabela 4: Erro médio absoluto (em metros), com desvio padrão - Fase 3.

Também podemos extrair das tabelas informações sobre a variação na acurácia de cada smartphone quando fixada uma técnica. Mas, para uma melhor visualização, foram criados os gráficos da Figura 8 para a fase 2 e da Figura 9 para a fase 3. Não foram criados gráficos para a primeira fase, pois os resultados são bastante similares para todos os cenários testados.

Na Figura 8-(A), podemos visualizar a diferença entre os resultados dos quatro smartphones

utilizando a técnica k-NN na fase 2. Para o k-NN, o melhor smartphone foi o modelo GT-S6500, com uma taxa de acerto de 99,49%, enquanto o pior foi o modelo HTC Wildfire, com uma taxa de acerto de 89,66%. A Figura 8-(B) tem o mesmo objetivo da Figura 8-(A), só que agora para a técnica *Random Forest*. Os smartphones nesse caso têm resultados mais semelhantes entre si do que no caso anterior. O modelo GT-S6500 foi novamente o smartphone com a melhor acurácia, só que dessa vez com uma taxa de acerto de 100%. Para este caso, o celular que teve a pior acurácia foi o LT22i, com 98,19%. Portanto, a diferença entre o telefone que mais acertou e o que mais errou foi 1,81%. A Figura 8-(C) é semelhante às duas anteriores, mas agora utilizando SVM. O smartphone GT-S5360 teve o melhor resultado, com 98,99% de taxa de acerto. Entretanto, a diferença dele para o GT-S6500, o segundo melhor, foi somente de 0,15%. Para este método, o HTC Wildfire foi o pior resultado, com 93,93% de acerto. Um fato importante a ser observado é que os modelos GT-S6500 e GT-S5360 ficaram entre os dois melhores resultados para as três técnicas, com resultados próximos entre si. Logo, entre os quatro smartphones testados, eles parecem ter o melhor desempenho mesmo com técnicas diferentes. Porém, a diferença na acurácia dos quatro aparelhos não aparenta ser muito crítica para os casos analisados nesta fase pois, para todas as técnicas, a média dos smartphones ficou acima de 90%.

Na Figura 9-(A), pode-se observar os erros médios absolutos de cada celular para o k-NN, na Fase 3. O modelo GT-S6500 foi o smartphone com o menor erro de localização, com 1,62 metros, enquanto o smartphone HTC Wildfire teve o maior erro: 7,76 metros. Os modelos GT-S5360 e LT22i apresentaram erros semelhantes. A Figura 9-(B) também mostra os erros dos smartphones, só que para a técnica *Random Forest*. Para este método, o celular que apresentou o melhor resultado foi, novamente, o GT-S6500, com 2,24 metros, e o que apresentou o pior foi, novamente, o HTC Wildfire, com 8,78 metros. É importante notar que a maioria dos aparelhos foram melhores no método anterior. A Figura 9-(C) apresenta as mesmas informações das figuras anteriores, mas utilizando o método SVM. Mais uma vez, observa-se que o modelo de smartphone que apresenta melhor resultado é o GT-S6500, e o pior é o HTC Wildfire. Porém, neste caso, a acurácia dos smartphones é pior em relação aos dois casos anteriores, com o melhor erro de localização sendo 3,56 metros e o pior sendo 10,39 metros. No contexto de localização *indoor*, até pequenos aumentos no erro de localização já causam problemas. Observa-se que, para os cenários executados, o modelo GT-S6500 apresenta o melhor resultado para todas as técnicas. Ele também é o modelo que sofre menos variação nos resultados quando as técnicas

Figura 8: Taxas de acerto de cada smartphone na fase 2, para cada uma das técnicas.

Fonte: Elaborado pela autora.

são variadas.

Com a finalidade de verificar se a diferença dos resultados entre os métodos ou smartphones utilizados são estatisticamente relevantes, foram realizados quatro testes de hipótese. O primeiro teste foi realizado com os três classificadores (k-NN, *Random Forest* e SVM) da fase 2. O segundo com os três regressores (k-NN, *Random Forest* e SVR) da fase 3. O objetivo dos dois testes iniciais era descobrir qual das técnicas obteve melhor acurácia em cada fase, levando em consideração os resultados encontrados por elas para todos os smartphones. O terceiro teste de hipótese foi realizado para comparar os resultados dos smartphones na fase 2 e o quarto para comparar os resultados dos aparelhos na fase 3. O objetivo dos dois últimos testes era descobrir se a acurácia de um smartphone para outro variava significativamente.

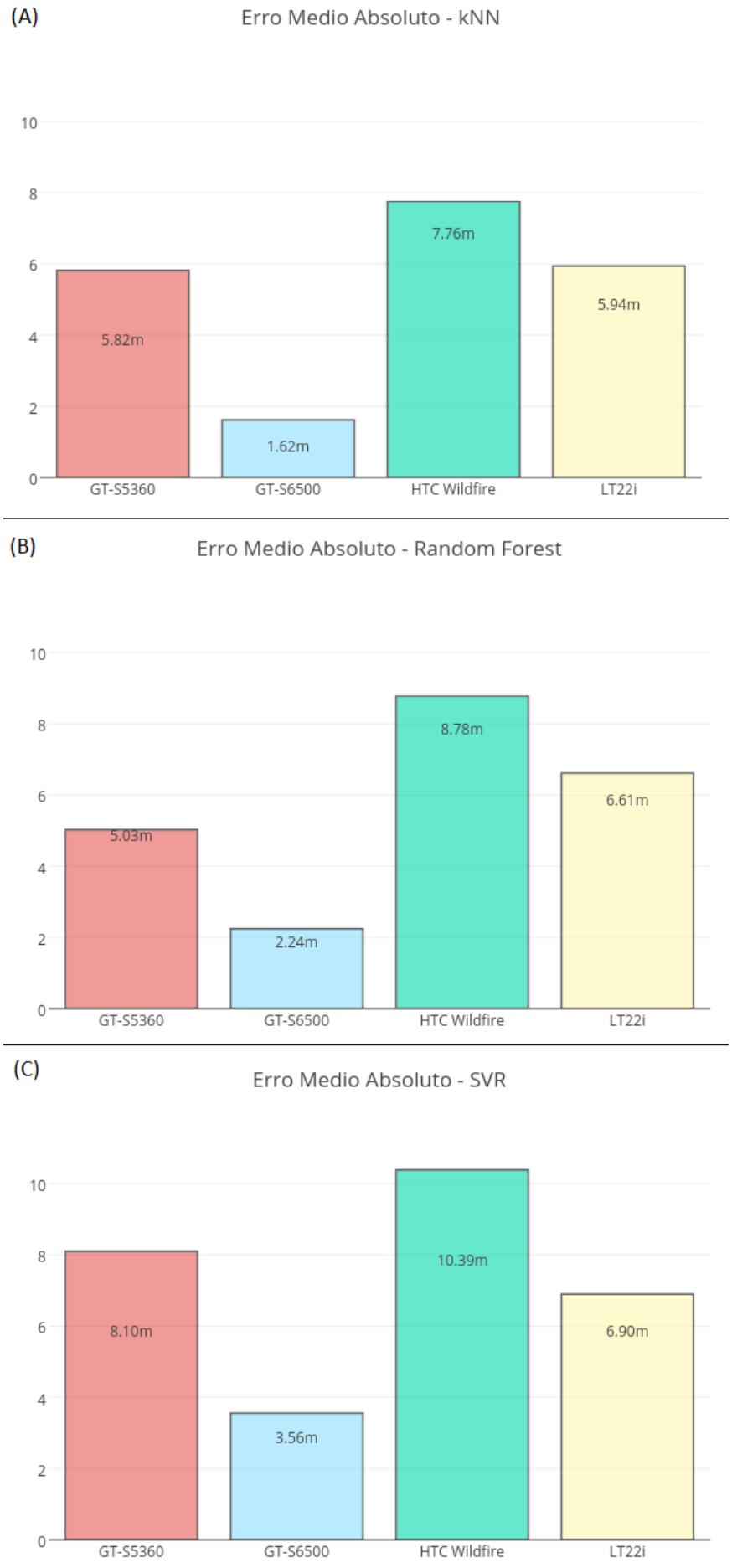
Os quatro testes seguiram a mesma metodologia. Primeiro foi realizado o teste de Shapiro-Wilk para saber se as amostras seguiam uma distribuição normal. Como nenhuma das amostras seguiu, foi necessário realizar um teste não-paramétrico. O teste escolhido foi o teste de Friedman [28], onde a hipótese nula H_0 era de que o desempenho das técnicas (ou smartphones) era equivalente e a hipótese alternativa H_1 era de que havia diferença no desempenho. A hipótese nula é rejeitada se o valor-p é menor do que um certo nível de significância α. Esse nível de significância é a probabilidade de se rejeitar a hipótese nula quando ela é verdadeira. Neste trabalho o valor adotado para α foi 0.05, pois este é o valor tradicionalmente utilizado. Já o valor-p mede o quão compatível as amostras são com a hipótese nula. Portanto, pequenos valores de p fazem a hipótese nula ser improvável.

Para todas as quatro configurações de teste mencionadas anteriormente, aplicando o teste de Friedman, a hipótese nula pôde ser rejeitada. Como H_0 foi rejeitada, o pós-teste de Nemenyi foi aplicado para fazer uma comparação par a par [29]. As Tabelas 5, 6, 7 e 8 apresentam os resultados da comparação par a par.

Na Tabela 5, são mostradas as comparações do valor-p para os três classificadores da fase 2. Pelos resultados, pode-se dizer que a *Random Forest* difere das outras técnicas porque seu valor-p é menor do que o nível de significância considerado. Por outro lado, não se pode dizer que a SVM e o k-NN são diferentes, pois o valor-p é maior que α.

Na Tabela 6, as mesmas comparações para o valor-p são feitas, só que agora para os regressores da fase 3. Para este caso, a *Random Forest* e o k-NN não podem ser considerados diferentes, pois o valor-p é 0,57. Já a *Random Forest* e a SVM podem ser consideradas distintas, assim como a SVM e o k-NN, pois seus valores de p são muito menores que 0,05.

Figura 9: Erro Médio Absoluto de cada smartphone na fase 3, para cada uma das técnicas.



Fonte: Elaborado pela autora.

Técnica 1	Técnica 2	valor-p
Random Forest	k-NN	5×10^{-5}
Random Forest	SVM	$1,3 \times 10^{-10}$
SVM	k-NN	$0,057$

Tabela 5: Teste de Nemenyi para os classificadores da fase 2.

Técnica 1	Técnica 2	valor-p
Random Forest	k-NN	$0,57$
Random Forest	SVM	$7,9 \times 10^{-6}$
SVM	k-NN	$3,6 \times 10^{-8}$

Tabela 6: Teste de Nemenyi para os regressores da fase 3.

As Tabelas 7 e 8 comparam o valor-p entre cada dupla de celulares para observar se, estatisticamente, há diferença nos resultados para diferentes tipos de aparelhos. A Tabela 7 se refere aos resultados dos smartphones na fase 2. Nela, nota-se que o único par de aparelhos que não podem ser considerados diferentes um do outro é o GT-S5360 com o GT-S6500, pois o valor-p nesse caso é maior que 0,05. Para todas as outras combinações, o valor-p é menor do que o nível de significância. Portanto, todos eles diferem entre si. Na Tabela 8 foram comparados os resultados dos celulares na fase 3. Nesta tabela, também só há um par de smartphones que não rejeita a hipótese nula. São eles: GT-S5360 e LT22i. Em todas as outras combinações, pode-se dizer que os resultados diferem.

Pelas análises feitas neste trabalho, conclui-se que os smartphones testados podem levar a localizações diferentes mesmo quando usada a mesma técnica de *fingerprinting*. Além disso, dentre os métodos utilizados, também pode-se dizer que a *Random Forest* foi a técnica que obteve a melhor generalização, pois classificou melhor os smartphones durante a fase 2 e obteve desempenho similar ao k-NN durante a fase 3.

Smartphone 1	Smartphone 2	valor-p
GT-S5360	GT-S6500	$6,2683 \times 10^{-1}$
GT-S5360	HTC Wildfire	$2,5 \times 10^{-10}$
GT-S5360	LT22i	$1,436 \times 10^{-2}$
GT-S6500	HTC Wildfire	6×10^{-14}
GT-S6500	LT22i	$1,6 \times 10^{-4}$
HTC Wildfire	LT22i	$1,81 \times 10^{-3}$

Tabela 7: Teste de Nemenyi para os smartphones - Fase 2.

Smartphone 1	Smartphone 2	valor-p
GT-S5360	GT-S6500	1×10^{-4}
GT-S5360	HTC Wildfire	$1,5 \times 10^{-5}$
GT-S5360	LT22i	$9,783 \times 10^{-1}$
GT-S6500	HTC Wildfire	$2,7 \times 10^{-14}$
GT-S6500	LT22i	$1,5 \times 10^{-5}$
HTC Wildfire	LT22i	1×10^{-4}

Tabela 8: Teste de Nemenyi para os smartphones - Fase 3.

4 Conclusão

Um dos principais problemas relacionados ao uso da técnica baseada em *fingerprinting* em localização *indoor* é que smartphones diferentes captam níveis de sinal Wi-Fi diferentes para um mesmo ponto de acesso. Dito isso, o presente trabalho apresentou uma análise comparativa de algoritmos de aprendizagem de máquina aplicados para a técnica de localização *indoor* baseada em *fingerprinting*, considerando a adoção de diferentes modelos de smartphones. Foram feitos experimentos para três técnicas diferentes com a finalidade de entender melhor a diferença na acurácia dos celulares para cada uma das técnicas. O trabalho também buscou encontrar dentre as técnicas utilizadas, qual teria a melhor capacidade de generalização. Esta última informação é de grande importância para tentar minimizar as diferenças nos resultados causadas pelos diferentes níveis de WiFi observados por cada smartphone.

Nos experimentos foram utilizados três algoritmos de aprendizagem de máquina: k-NN, *Random Forest* e SVM. Os algoritmos foram treinados para encontrar o prédio em que o usuário estava, o andar e as coordenadas geográficas. Para isso, utilizou-se dois classificadores, para encontrar o prédio e andar, e dois regressores, para encontrar a latitude e a longitude. Observou-se que embora houvesse diferenças nas acurácias dos classificadores de cada celular, eles ainda assim geravam bons resultados para o problema de localização. Entretanto, para encontrar as coordenadas, os celulares tiveram resultados mais divergentes. Para a SVR, por exemplo, houve uma diferença de quase 7 metros entre os celulares com melhor e pior resultado.

Os resultados para cada uma das três técnicas também foram comparados. Observou-se que para a classificação, o *Random Forest* obteve as melhores acurácias com qualquer um dos celulares utilizados. Na classificação seu desempenho foi similar ao do k-NN. Já a SVM obteve os piores resultados quando comparada às outras duas técnicas.

Por fim, pôde-se confirmar que, de fato, a diferença nos sinais de WiFi percebidos por cada smartphone gera uma diferença, as vezes preocupante, nos resultados obtidos pelo *fingerprinting*. Por esse motivo, muitos métodos de localização estão sendo propostos, incluindo combinações do *fingerprinting* com outras técnicas [7]. Porém, o *fingerprinting* ainda têm várias vantagens se comparado às soluções existentes. Por isso uma extensão deste trabalho pode ser realizada a fim de comparar mais celulares e mais técnicas, com o objetivo de tentar descobrir algoritmos que consigam generalizar melhor, obtendo assim bons resultados para todo tipo de smartphone.

Referências

[1] J. Torres-Sospedra, R. Montoliu, A. Martinez-Uso et al. "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems". In *2014 International Conference on Indoor Positioning and Indoor Navigation*, pages 261–270, Busan, South Korea, 2014.

[2] H. Liu, H. Darabi, P. Banerjee, and J. Liu. "Survey of Wireless Indoor Positioning Techniques and Systems". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[3] S. Bozkurt, G. Elibol, S. Gunal, and U. Yayan. "A Comparative Study on Machine Learning Algorithms for Indoor Positioning". In *Innovations in Intelligent SysTems and Applications (INISTA)*, Madrid, Spain, 2015.

[4] G. Lui, T. Gallagher, B. Li, A. G. Dempster, and C. Rizos. "Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of WLAN localization". In *2011 International Conference on Localization and GNSS*, pages 53–57, Tampere, Finland, 2011.

[5] A. K. M. M. Hossain, H. N. Van, Y. Jin, and W. S. Soh. "Indoor Localization Using Multiple Wireless Technologies". In *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*, pages 1–8, Pisa, Italy, 2007.

[6] R. D. A. Timoteo, L. N. Silva, D. C. Cunha, and G. D. C. Cavalcanti. "An approach using support vector regression for mobile location in cellular networks". *Computer Networks*, 95:51–61, 2016.

[7] P. A. Bello. "Localização Indoor: dos Princípios ao Crowdsourcing". Master's thesis, Universidade Federal de Pernambuco, 2015.

[8] S. Lee, K. N. Ha, and K. C. Lee. "A Pyroelectric Infrared Sensor-based Indoor Location-Aware System for the Smart Home". *IEEE Transactions on Consumer Electronics*, 52, 2006.

[9] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Pati. "LANDMARC: Indoor Location Sensing Using Active RFID". *Wireless Networks*, 10(6):701–710, 2004.

[10] A. Ward, A. Jones, and A. Hopper. "A new location technique for the active office". *IEEE Personal Communications*, 4:42–47, 1997.

[11] Z. Farid, R. Nordin, and M. Ismail. "Recent Advances in Wireless Indoor Localization Techniques and System". *Journal of Computer Networks and Communications*, vol. 2013:12, 2013.

[12] P. Bahl and V. N. Padmanabhan. "RADAR: an in-building RF-based user location and tracking system". In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, pages 775–784, Tel Aviv, Israel, 2000.

[13] M. Youssef. "The Horus WLAN location determination system". In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218, Seattle, Washington, 2005.

[14] J. Lee, B. Choi, and E. Kim. "Novel range-free localization based on multidimensional support vector regression trained in the primal space". *IEEE Transactions on Neural Networks and Learning Systems*, 24(7):1099–1113, 2013.

[15] L. Calderoni, M. Ferrara, A. Franco, and D. Maio. "Indoor localization in a hospital environment using Random Forest classifiers". *Expert Systems with Applications*, 42(1):125–134, 2015.

[16] T. M. Mitchel. *"Machine Learning"*, pages 230–231. McGraw-Hill Book Company, 1 edition, 1997.

[17] J. Calvo-Zaragoza, J. J. Valero-Mas, and J.R. Rico-Juan. "Improving kNN multi-label classification in Prototype Selection scenarios using class proposals". *Pattern Recognition*, 48(5):1608–1622, 2015.

[18] T. Hastie , R. Tibshirani, and J. Friedman. *"The Elements of Statistical Learning"*, pages 587–596. Springer, 2 edition, 2009.

[19] G. James, D. Witten, T. Hastie, and R. Tibshirani. *"An Introduction to Statistical Learning"*, pages 316–321. Springer, 1 edition, 2013.

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

[21] A. Lima. "Máquinas de Vetores Suporte na Classificação de Impressões Digitais". Master's thesis, Universidade Federal do Ceará, 2002.

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

[23] V. Vapnik, S. E. Golowich, and A. Smola. "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing". *Advances in Neural Information Processing Systems 9*, pages 281–287, 1997.

[24] T. B. Trafalis and H. Ince. "Support vector machine for regression and applications to financial forecasting". In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, pages 348–353, Como, Italy, 2000.

[25] A. Smola and B. Scholkopf. "A Tutorial on Support Vector Regression". *Statistics and Computing*, 14:199–222, 2003.

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

[27] T. Vincenty. "Direct and Inverse Solutions of Geodesics on the Ellipsoid With Application of Nested Equations". *Survey Review*, 23(176):88–93, 1975.

[28] M. Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[29] J. Demsar. "Statistical Comparisons of Classifiers over Multiple Data Sets". *Journal of Machine Learning Research*, 7:1–30, 2006.