Relatório projeto final - Introdução ao Aprendizado de Máquina

Localização Indoor utilizando algoritmos de aprendizado de máquina a partir de medidas de RSSI

Elço João dos Santos Junior
Rodrigo Kobashikawa Rosa

Dezembro 2018

1 Introdução

Atualmente, sistemas que geram informações de posicionamento tem se tornado de grande valia, visto que podem ser aplicados a diversos tipos de problemas, como por exemplo, localização de pessoas, insumos, ativos e produtos. Em ambientes *outdoor*, soluções como o *Global Positioning System* (GPS) apresentam boa acurácia ao determinar o posicionamento de um objeto, porém, o mesmo não pode ser afirmado para ambientes *indoor*, no qual os resultados apresentam baixa precisão principalmente devido à atenuação sofrida pelos sinais de satélite por conta do concreto, telhados e paredes.

Portanto, outras formas de se realizar o *tracking* em ambientes *indoor* se fazem necessárias. Técnicas já conhecidas, tais como: triangulação, trilateração, *time of arrival (ToA)* e Min-Max, utilizam valores de potência, distância, tempo de propagação e modelagem de canal [1,2,3,4] para definir a posição de um determinado objeto. Apesar de serem soluções já aceitas e utilizadas na prática, estas quando não necessitam de um *hardware* complexo, apresentam uma baixa acurácia ao determinar a posição, visto que conversões de medidas de RSSI (*Received Signal Strength Indication*) em distância possuem baixa precisão [5], pois estas medidas possuem grandes oscilações conforme o canal de comunicação sem fio varia. Portanto, à uma mesma distância, observaria-se variação de alguns metros nas medidas, resultando em estimativas de localização não confiáveis.

Devido a isso, seria interessante estudar uma outra abordagem para realizar localização em ambientes *indoor*. Uma opção seria implementar algoritmos de *machine learning* para realizar a modelagem do canal de forma dinâmica baseando-se em dados rotulados utilizados anteriormente para treinamento, como discutido em [6] e [7].

2 Conjuntos de dados utilizados

Os trabalhos apresentados em [8] e [9] utilizam o conjunto de dados *UjiIndoorLoc*, disponível no *Kaggle* [13], para realizar o estudo de técnicas de aprendizado de máquina para efetuar a tarefa de localização. Por ser um *dataset* bem aceito pela literatura, este foi um dos conjuntos de dados utilizados no projeto.

O *UjiIndoorLoc*, reúne valores de RSSI de pontos de acesso de Wi-Fi, latitude, longitude, andar e foi coletado em 3 diferentes prédios na Universitat Jaume I. O *dataset* já está separado em conjunto de treinamento e validação, e foi utilizado como base para a competição IPIN2015.

Uma outra opção de *dataset*, também disponível no *Kaggle* [14], utilizou dispositivos *Bluetooth Low Energy* denominados *Beacons* para coletar medidas de RSSI em um ambiente *indoor* que foi dividido em diferentes áreas. Esse conjunto de dados se torna interessante, inicialmente, em uma abordagem de classificação, onde poderia-se realizar predições para indicar em qual das regiões o objeto ou pessoa está, com

determinada probabilidade de acerto. Está disponível também uma planta baixa do ambiente com as áreas destacadas, a qual será mostrada posteriormente.

Medidas de RSSI obtidas por dispositivos *Bluetooth* para localização *indoor* vem sendo analisadas, como mostram os trabalhos [10, 11, 12]. Porém, não encontrou-se artigos que uniam estes tipos de medidas oriundas de tais dispositivos e técnicas de *machine learning*. Devido a isso, o conjunto de dados em questão poderia ser utilizado buscando uma contribuição científica de maior destaque.

3 Trabalhos relacionados

Algoritmos de *machine learning* vem sendo estudados para realizar a tarefa de localização *indoor*.

- Em Salamah et al. [6] é comparado o uso de KNN, *Decision Tree*, SVM e classificadores *Random Forest* para métodos de localização indoor via RSSI *fingerprints* (Wi-Fi).
- Nos artigos [7,8] também são apresentados comparações entre algoritmos de *Machine Learning* utilizando o conjunto de dados *UjiIndoorLoc*. Em [7] concluí-se o poder do KNN para a localização *indoor*, sendo já muito utilizado para a função. Porém, novos estudos mostram que a utilização de redes neurais profundas [8] apresenta resultados tão bons ou até melhores.

4 Objetivos

O trabalho tem como objetivo estudar e comparar algoritmos de aprendizado de máquina de treinamento supervisionado para realizar a tarefa de localização em ambientes *indoor* a partir de valores de RSSI de dispositivos *bluetooth*, buscando atingir valores de acurácia próximos aos apresentados na literatura.

Os algoritmos propostos a serem avaliados são o k-NN (*K-Nearest-Neighbor*), SVM (*Support Vector Machines*), *Random Forest*, Regressão Linear Multivariada e *Deep Neural Nets*.

5 Resultados

Nesta seção, serão apresentados os resultados obtidos utilizando ambos *datasets* que foram selecionados para o trabalho.

5.1 *Dataset Bluetooth*

A Figura 1 representa o ambiente no qual as medidas foram realizadas. O pontos em verde indicam a posição dos *beacons*, totalizando 13 dispositivos que foram utilizados para transmitir dados periódicos. Com um outro dispositivo, como por exemplo um *smartphone*, leu-se os valores de potência (RSSI) de cada beacon em cada um dos quadrados da imagem (por exemplo, posição O2, P3, Q5 etc). Para os *beacons* que não eram lidos em determinada posição devido a fatores como distância, o valor de -200 dBm foi atribuído.

Nota-se que em algumas regiões há a presença de paredes (A1, U10, N11 etc), as quais não estarão presentes no conjunto de dados. Além disso, as medidas de RSSI não foram coletadas em todas as regiões disponíveis, resultando em 105 regiões disponíveis para predição.

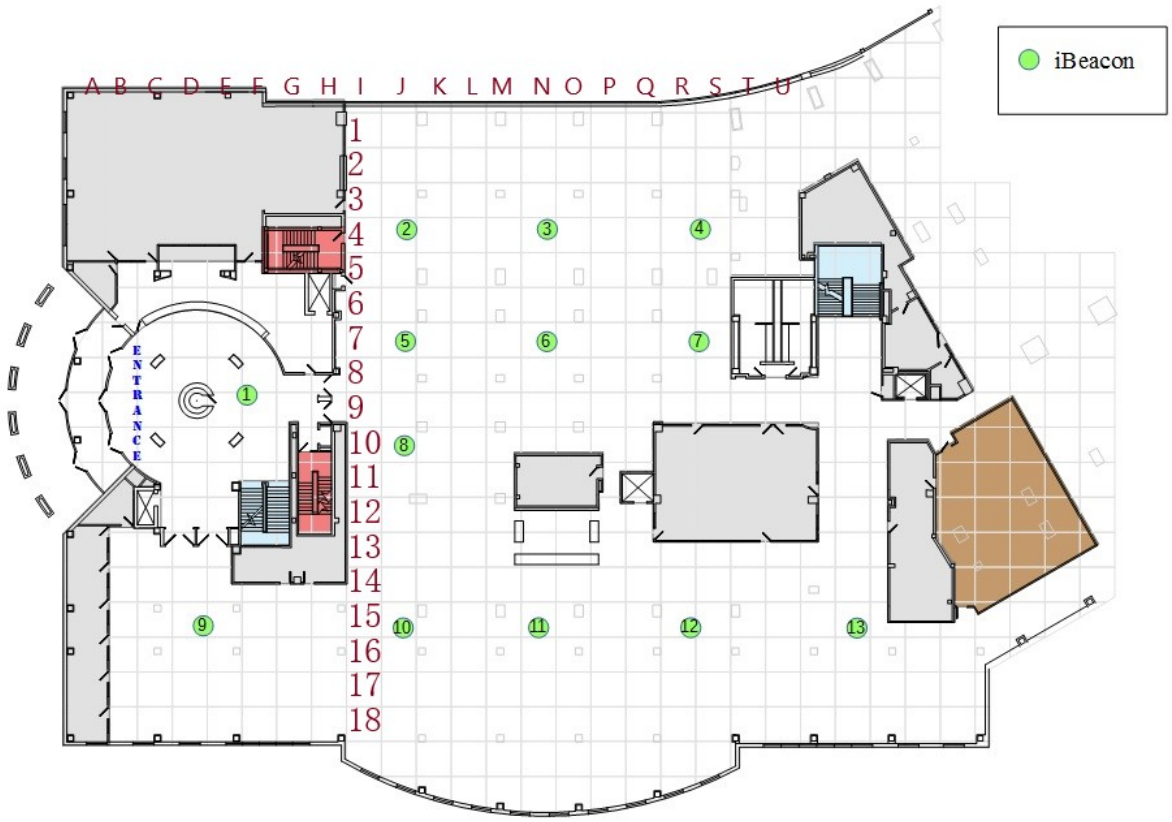


Figura 1: Planta baixa do ambiente mapeado.

A Figura 2 mostra uma representação do ambiente destacando em cores as regiões nas quais medidas de RSSI foram coletadas. As regiões em branco são aquelas que não foram mapeadas, enquanto que onde haviam paredes, os quadrados foram pintados de preto e branco.

As oito primeiras linhas do *dataset* resultante após as medidas podem ser visualizadas na Figura 3. Nota-se que há 13 atributos (RSSI de cada *beacons*) e 1 variável *target* que é a localização no ambiente. A informação de data e hora não foi utilizada. Um total de 1420 medidas foram realizadas, as quais foram divididas em 70% para treinamento e 30% para teste.

A metodologia escolhida para obter os resultados foi iniciar os modelos com parâmetros escolhidos baseando-se na literatura. Em [15], realizou-se uma comparação entre os algoritmos *Random Forest* e k-NN, obtendo-se acurácias de 91% e 64.26%, respectivamente. Em [16], uma análise usando SVM foi realizada e com os parâmetros selecionados obteve-se uma acurária com poucos metros de erro.

Após utilizar os parâmetros base dos trabalhos citados acima, realizou-se a técnica de *Grid Search* com *Cross Validation* em torno destes valores para obter parâmetros otimizados e aumentar a acurácia.

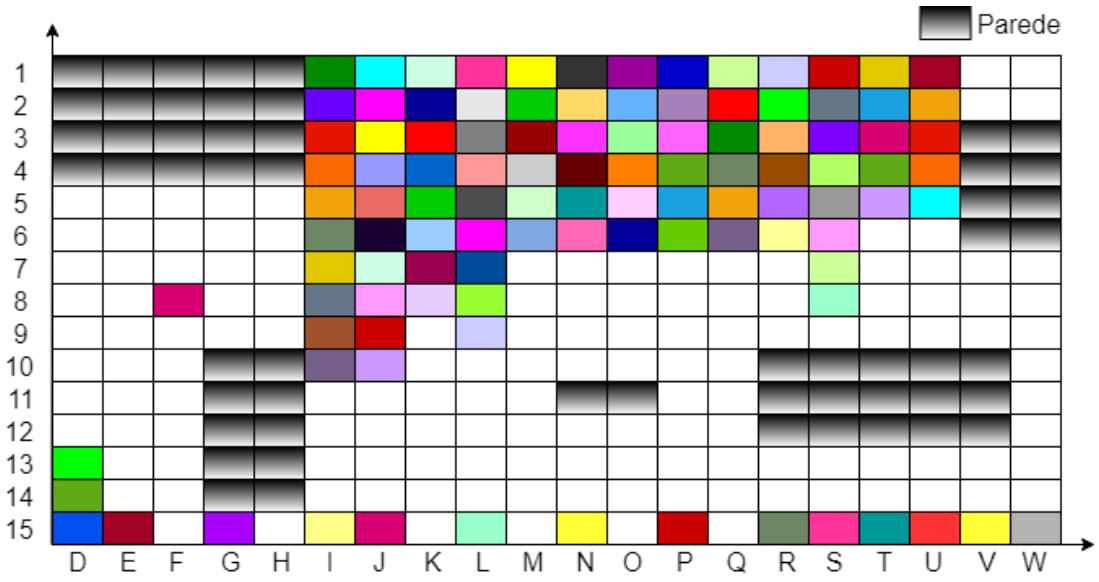


Figura 2: Representação que mostra regiões mapeadas.

	location	date	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
0	O02	10-18-2016 11:15:21	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
1	P01	10-18-2016 11:15:19	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
2	P01	10-18-2016 11:15:17	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
3	P01	10-18-2016 11:15:15	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
4	P01	10-18-2016 11:15:13	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
5	P01	10-18-2016 11:15:11	-200	-200	-82	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200
6	P01	10-18-2016 11:15:09	-200	-200	-80	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
7	P02	10-18-2016 11:15:07	-200	-200	-86	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200

Figura 3: Conjunto de dados.

5.1.1 Classificação no *dataset* original

O primeiro passo foi, utilizando as técnicas de classificação escolhidas no trabalho, gerar modelos sobre o conjunto de dados original. Os resultados podem ser visualizados na Tabela 3.

Método	Acurácia (%)
SMV	29.34
Random forest	27.70
k-NN	27.46

Tabela 1: Resultados com *dataset* padrão.

Este teste inicial mostrou resultados não tão satisfatórios, com uma acurácia máxima de 29.34% utilizando SVM.

5.1.2 Classificação separando os eixos

Uma outra opção que se encontrou foi separar os rótulos em dois eixos, um contendo letras e o outro contendo números. Realizando a mesma comparação do item anterior, obteve-se os resultados mostrados na tabela abaixo.

Método	Acurácia eixo X (%)	Acurácia eixo Y (%)
SVM	43.66	51.88
Random forest	44.6	46.94
k-NN	42.25	47.18

Tabela 2: Resultados com *dataset* separando em eixos.

Ao utilizar a técnica de separar a região em eixos, obteve-se um aumento da acurácia de predição, como esperado. Para o eixo X, temos 44.6% usando Random Forest e para o eixo Y temos 51.88% usando SVM. Apesar de a acurácia ter aumentado, é preciso que ambos modelos acertem para que a posição seja acertada de fato. No geral, o SVM apresenta maior probabilidade de acerto.

5.1.3 Dobrando a dimensão de cada região

Nesta etapa do trabalho, dobrou-se a área de cada região mapeada para localização. A Figura 4 mostra uma representação do espaço após realizar tal tarefa. Observa-se agora um total de 37 regiões possíveis para predição.

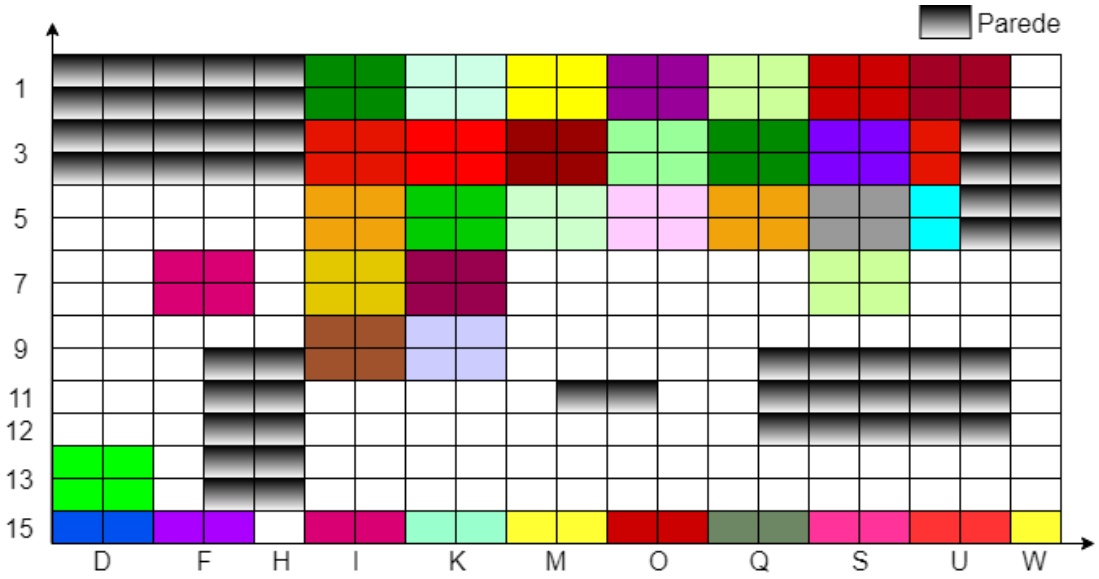


Figura 4: Representação que mostra regiões mapeadas.

Executando novamente os classificadores, porém agora sobre este conjunto de dados modificado, obteve-se os resultados mostrados abaixo.

Neste teste, obteve-se uma acurácia máxima de 44.37% usando o algoritmo de k-NN, resultado 1.53 vezes maior do que no primeiro teste.

Método	Acurácia (%)
SMV	43.42
Random forest	42.72
k-NN	44.37

Tabela 3: Resultados com *dataset* modificado.

5.1.4 Quadruplicando a dimensão de cada região

Aumentou-se cada região em quatro vezes, resultando na representação mostrada na Figura 5. O número de regiões é reduzido a 16, porém estas passam a cobrir uma área maior no ambiente *indoor*.

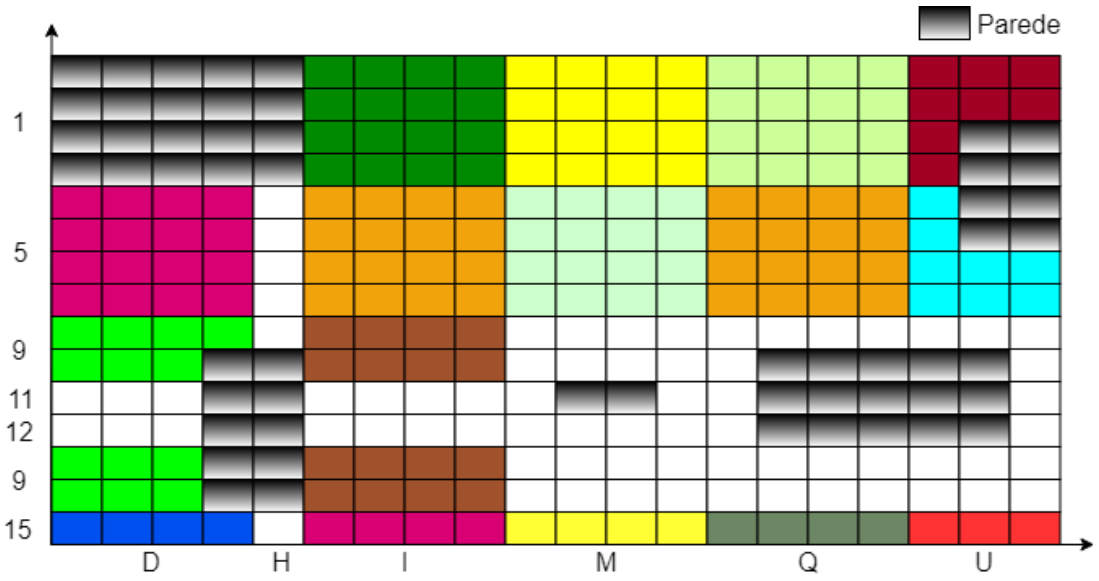


Figura 5: Representação que mostra regiões mapeadas.

Usando esta abordagem, obteve-se as acurácias mostradas na Tabela 4.

Método	Acurácia (%)
SMV	70.18
Random forest	72.06
k-NN	68.78

Tabela 4: Resultados com *dataset* modificado.

Quadruplicando cada região, obteve-se uma acurácia máxima de 72.06% usando o algoritmo de *Random Forest*, resultado 2.5 vezes maior do que no primeiro teste.

No geral, aumentar o tamanho de cada área no ambiente *indoor* aumenta a acurácia de predição do modelo, como era de se esperar. Em muitas aplicações, conhecer a posição aproximada de uma pessoa ou objeto já é suficiente para atender certos requisitos, portanto, adotar regiões maiores com um maior número de leituras de RSSI é benéfico.

5.1.5 Regressão Linear Multivariada

A próxima etapa do projeto foi enxergar o problema de uma outra perspectiva, interpretando-o como uma tarefa de regressão, e não de classificação. Deste modo, cada *label* de localização foi convertida para um número real (ficando em uma faixa de 0 a 104), correspondendo à variável y do modelo. Cada leitura

de RSSI corresponde a um parâmetro de entrada, de modo que a equação que representa o problema será dada por:

$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_{13} x_{13}$$

A tarefa é, portanto, encontrar os valores de ω_n que melhor se encaixam ao problema.

A Figura 6 mostra os valores de RSSI normalizados para as regiões mapeadas no ambiente, levando em consideração o *beacon* 3. Valores de RSSI nulos indicam que naquela determinada região o dispositivo não foi lido.

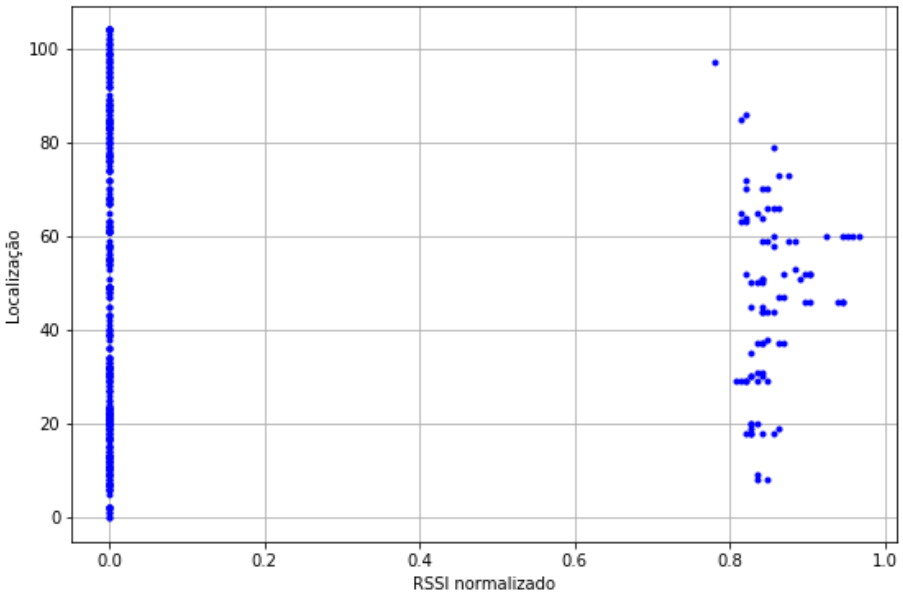


Figura 6: Valores de RSSI do *beacon* 3 para diferentes regiões.

Utilizando a biblioteca de regressão linear do *sklearn*, realizou-se o treinamento sobre 70% do conjunto e predições sobre 30%. Para analisar a qualidade do modelo, calculou-se as métricas de desempenho *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE) e *R square* (R^2). Os resultados estão mostrados na Tabela 5.

Métrica	Valor
MAE	10.86
MSE	178.98
RMSE	13.38
R^2	0.79

Tabela 5: Métricas de desempenho do modelo de regressão.

As métricas observadas acima indicam um resultado positivo, mostrando que o modelo construído realiza um trabalho satisfatório. Parâmetros como o R^2 *score* possuem um valor máximo de 1, portanto o valor 0.79 obtido é razoavelmente bom. Além disso, a métrica RMSE também demonstra a qualidade do modelo, pois, ao analisar seu valor dentro da faixa de valores da variável alvo (posição, de 0 a 104), o resultado de 13.4 é excelente.

A Figura 7 mostra o modelo obtido referente ao *beacon* 5.

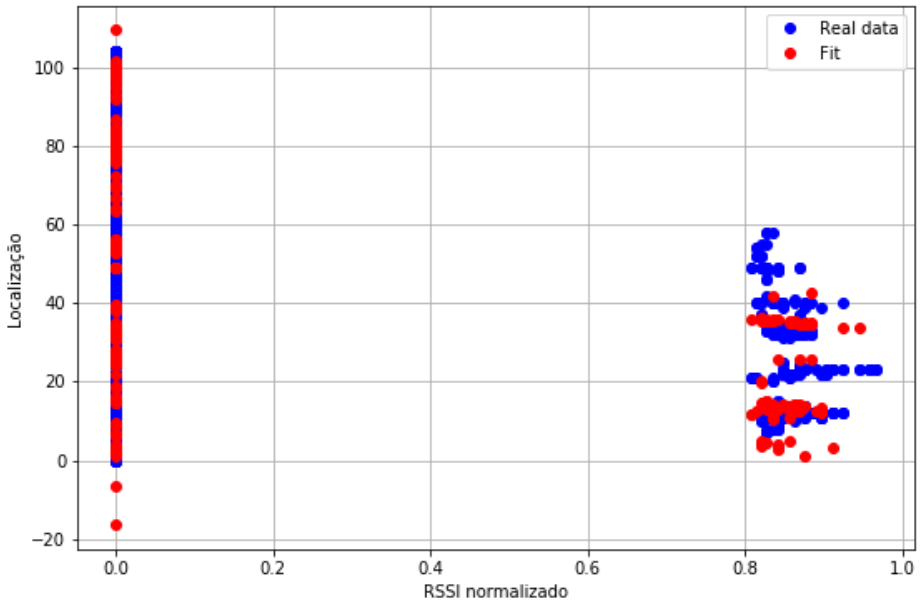


Figura 7: Valores reais e modelados do *beacon* 5 para diferentes regiões.

5.2 *Dataset UjiIndoorLoc*

O *dataset* contém 529 atributos sendo que as primeiras 520 correspondem aos valores de RSSI obtidos dos WLAN e as 9 restantes são informações sobre posicionamento (latitude, longitude, andar, prédio, ID do espaço, dentro ou fora do prédio) e também informações sobre como foi feita a medida (usuário, telefone, data e hora). No total são 21049 amostras coletadas, separadas em conjunto de treinamento (19937) e validação (1111). Uma representação do *dataset* pode ser visto na Figura 8.

WAP004	WAP005	WAP006	WAP007	WAP008	WAP009	WAP010	...	WAP520	LONGITUDE	LATITUDE	FLOOR	BUILDINGID	SPACEID	RELATIVEPOSITION
100	100	100	100	100	100	100	...	100	-7541.2643	4.864921e+06	2	1	106	2
100	100	100	100	100	100	100	...	100	-7536.6212	4.864934e+06	2	1	106	2
100	100	100	100	-97	100	100	...	100	-7519.1524	4.864950e+06	2	1	103	2
100	100	100	100	100	100	100	...	100	-7524.5704	4.864934e+06	2	1	102	2
100	100	100	100	100	100	100	...	100	-7632.1436	4.864982e+06	0	0	122	2

Figura 8: Conjunto de dados.

Como comentado na seção 2, esse *dataset* utiliza medidas de RSSI captados por WLANs coletados em 3 prédios diferentes da Universitat Jaume I, apresentados na Figura 9. Cada prédio possuí 4 ou 5 andares e possuem 123 ambientes diferentes classificados no conjunto de treinamento.

5.2.1 Análise dos dados

O objetivo utilizando esse conjunto de dados é de prever a localização *indoor* através das coordenadas de latitude e longitude disponíveis, portanto foi separado o conjunto de dados por prédio e andar.

Observando a Tabela 6, percebe-se que apenas o prédio 2 possuí o andar 4 e que a maior quantidade de amostras se encontram nos andares 1 e 2 dos prédios. Por esse motivo nas analises de desempenho serão utilizadas os conjuntos de dados desses andares, marcados em negrito na tabela.

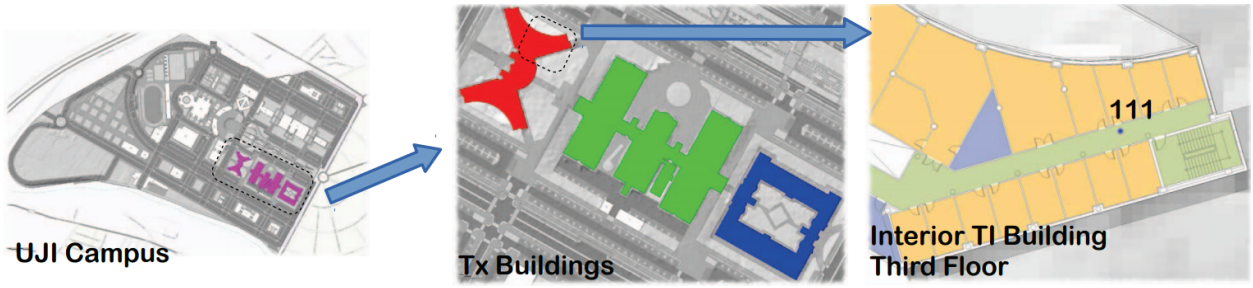


Figura 9: Mapa do campus e zoom nos 3 prédios.

Prédio	Andar	Amostras de treinamento	Amostras de validação
0	0	1059	78
0	**1**	**1356**	**208**
0	**2**	**1443**	**165**
0	3	1391	85
1	0	1368	30
1	**1**	**1484**	**143**
1	2	1396	87
1	3	948	47
2	0	1942	24
2	**1**	**2162**	**111**
2	2	1577	54
2	3	2709	40
2	4	1102	39

Tabela 6: Amostras de treinamento e validação por andar do prédio.

5.2.2 Pré-processamento dos dados

Para o preprocessamento dos atributos foi substituído no conjunto as amostras de RSSI com valor de 100 para -110, um valor um pouco menor do que o mínimo medido no dataset (-104), e aplicado normalização min-max para escalar os dados na faixa entre 0 e 1.

Os valores das coordenadas, *targets* a serem preditos, foram subtraídos do menor valor respectivo a fim de converter em valores relativos de distância. Podendo assim ser obtido uma ideia do tamanho dos prédios.

Para o uso da rede convolucional foi feito um remodelamento dos atributos para poderem ser representados como uma figura 2D de dimensões 20x26

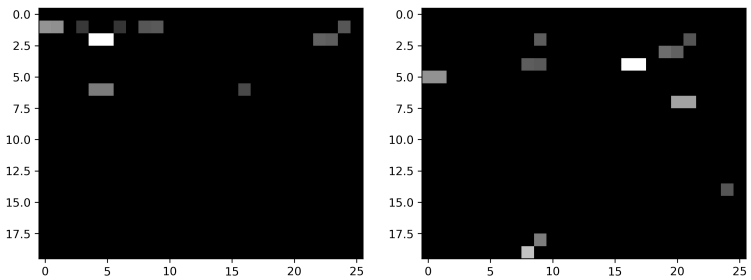


Figura 10: Exemplos do remodelamento em 2D dos atributos.

5.2.3 Métodos de regressão utilizados

Inicialmente foi feita uma comparação entre os algoritmos padrões utilizados para localização *indoor* como SVM, *Random Forest* e k-NN, em suas versões de regressão do *Scikit learn*. Utilizando o *GridSearchCV* foi feita uma procura dos hiperparâmetros e analisado as métricas de desempenho *mean absolute erro (MAE)*, *mean squared error (MSE)*, *root mean squared error (RMSE)* e *R square* (R^2).

Após essa comparação inicial, procurou-se utilizar redes neurais profundas como também redes convolucionais para observar como elas se comparam com os algoritmos anteriores.

O modelo de rede neural profunda como alguns passos buscaram inspiração em Bokzurt et al.[8] buscando reproduzir e expandir os resultados obtidos, Figura 12.

Consiste de 3 camadas *fully-connected* ocultas com 500 unidades com uma camada de *dropout* de 50% para evitar *overfitting*. A função de ativação utilizada em cada camada oculta foi a ReLU, sendo uma boa função *default* a ser utilizada. A função perda utilizada foi a MSE.

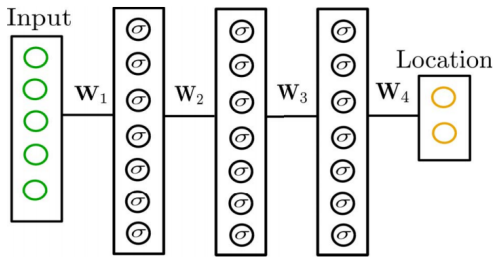


Figura 11: Estrutura da rede neural.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 500)	260500
activation (Activation)	(None, 500)	0
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 500)	250500
activation_1 (Activation)	(None, 500)	0
dropout_1 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 500)	250500
activation_2 (Activation)	(None, 500)	0
dropout_2 (Dropout)	(None, 500)	0
dense_3 (Dense)	(None, 2)	1002

Total params: 762,502
Trainable params: 762,502
Non-trainable params: 0

Figura 12: Modelo utilizado da rede neural.

O modelo de rede convolucional adotada para testes foi baseada em arquiteturaas de redes convolucionais

conhecidas como a LeNet e a AlexNet. A ideia de se utilizar a rede convolucional é de que transformando os dados como uma imagem os filtros conseguem extrair diversas características criando correlações e uma previsão melhor.

O primeiro modelo utilizado consiste de 2 camadas convolucionais, sendo a primeira com 64 filtros com *kernel size* 3x3 e a segunda com 128 filtros e *kernel size* 2x2 passando por uma camada de *maxpooling* de tamanho 2x2 para reduzir pela metade os parâmetros e uma camada de *dropout* para evitar *overfitting*. Logo em seguida, é transformado num array 1D e passa por 2 camadas *fully-connected* com 300 unidades cada.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 20, 26, 64)        640
_________________________________________________________________
activation (Activation)      (None, 20, 26, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 20, 26, 128)       32896
_________________________________________________________________
activation_1 (Activation)    (None, 20, 26, 128)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 10, 13, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 10, 13, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 16640)             0
_________________________________________________________________
dense (Dense)                (None, 300)               4992300
_________________________________________________________________
activation_2 (Activation)    (None, 300)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 602
=================================================================
Total params: 5,026,438
Trainable params: 5,026,438
Non-trainable params: 0
_________________________________________________________________
```

Figura 13: Modelo utilizado da rede convolucional 1.

O segundo modelo utilizado consiste de 4 camadas convolucionais com *kernel size* 2x2. As duas primeiras possuem 64 filtros e as duas consequentes possuem 128 filtros, logo após passam pelo mesmo processo de *maxpooling* e *dropout* e duas camadas ocultas *fully-connected*, porém estas agora com apenas 128 unidades.

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 20, 26, 64)	320
activation_3 (Activation)	(None, 20, 26, 64)	0
conv2d_3 (Conv2D)	(None, 20, 26, 64)	16448
activation_4 (Activation)	(None, 20, 26, 64)	0
conv2d_4 (Conv2D)	(None, 20, 26, 128)	32896
activation_5 (Activation)	(None, 20, 26, 128)	0
conv2d_5 (Conv2D)	(None, 20, 26, 128)	65664
activation_6 (Activation)	(None, 20, 26, 128)	0
conv2d_6 (Conv2D)	(None, 20, 26, 128)	65664
activation_7 (Activation)	(None, 20, 26, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 10, 13, 128)	0
dropout_2 (Dropout)	(None, 10, 13, 128)	0
flatten_1 (Flatten)	(None, 16640)	0
dense_2 (Dense)	(None, 128)	2130048
dense_3 (Dense)	(None, 128)	16512
activation_8 (Activation)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 2)	258

Total params: 2,327,810
Trainable params: 2,327,810
Non-trainable params: 0

Figura 14: Modelo utilizado da rede convolucional 2.

5.2.4 Prédio 0 - Andar 1

Para o andar 1 do prédio 0 foi utilizado o conjunto de treinamento/validação da Figura 15. Percebe-se como a forma da figura representa muito bem o prédio do campus representado em vermelho na Figura 9.

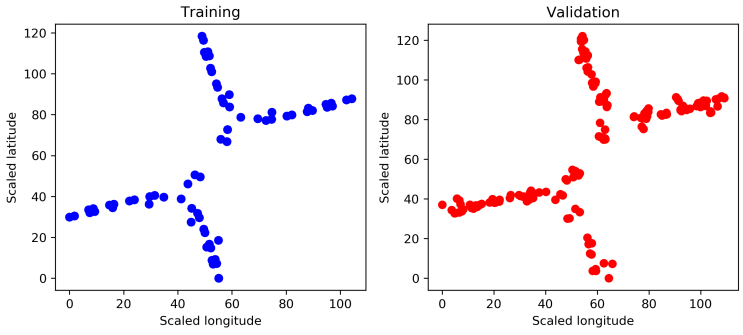


Figura 15

Comparando os resultados entre KNN, SVM e *RandomForest* após o *GridSearchCV* na Tabela 7, pode ser observado que o KNN apresentou os melhores resultados sendo que quanto menor o valor de erro é melhor e o score R2 tem seu melhor valor sendo 1, portanto valores próximos a ele são muito bons.

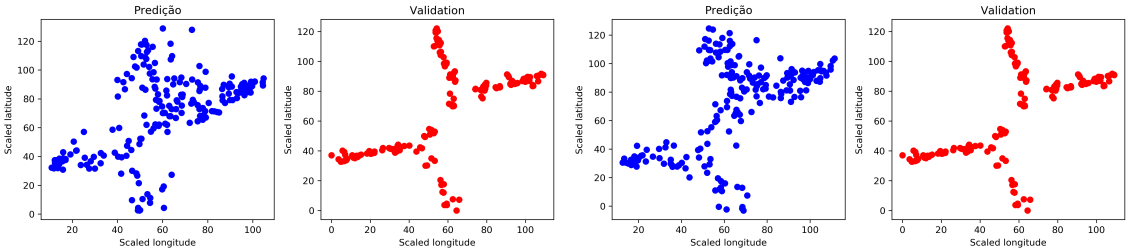
Métrica	**SVM**	**RandomForest**	**KNN**
Mean absolute error (MAE)	11.63	6.47	4.59
Mean squared error (MSE)	221.32	86.52	35.31
Root mean squared error (RMSE)	14.88	9.30	5.94
R square (R^2)	0.663	0.894	0.957

Tabela 7: Comparação das métricas de desempenho dos algoritmos SVM, RandomForest e KNN.

Utilizando a rede neural profunda e as redes convolucionais propostas foram obtidos resultados inferiores ao KNN e *RandomForest*, vistos na Tabela 8. Na figura 16 pode ser visto como para esse conjunto de dados a rede convolucional obteve um resultado melhor em relação à rede neural profunda mantendo a forma melhor e predições menos espalhadas.

Métrica	**DNN**	**CNN1**	**CNN2**
Mean squared error (MSE)	138.58	89.32	97.73
Root mean squared error (RMSE)	9.074	8.038	8.51
R square (R^2)	0.737	0.828	0.812

Tabela 8: Comparação das métricas de desempenho da DNN e das CNNs.



(a) Predição utilizando modelo com rede neural profunda.

(b) Predição utilizando modelo com rede convolucional

Figura 16: Comparativo das predições entre DNN e CNN.

5.2.5 Prédio 1 - Andar 1

Para o andar 1 do prédio 1 foi utilizado o conjunto de treinamento/validação da Figura 17.

Comparando os resultados entre KNN, SVM e *RandomForest* após o *GridSearchCV* na Tabela 9, diferente do caso do prédio 0, o SVM obteve resultado melhor do que o *RandomForest*. O KNN ainda foi o melhor dentre os três.

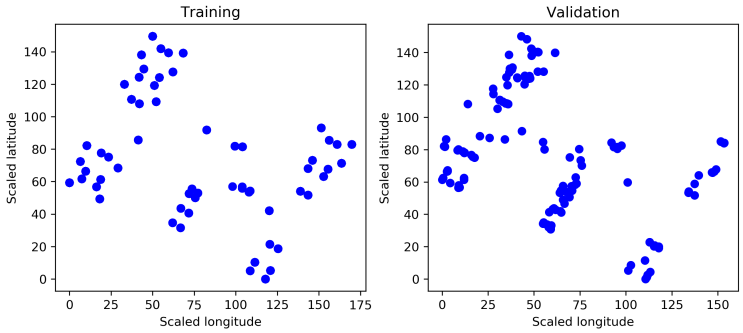


Figura 17

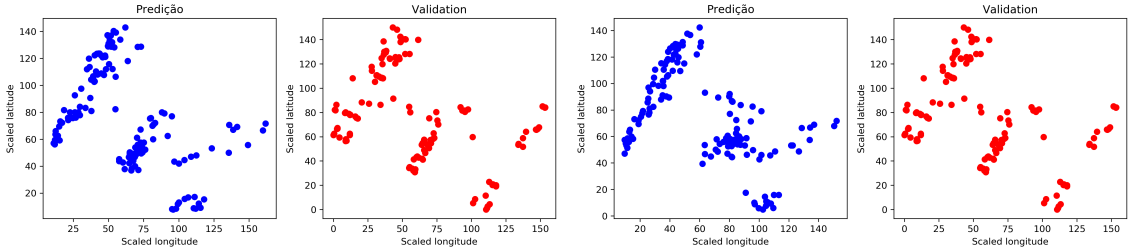
Métrica	SVM	RandomForest	KNN
Mean absolute error (MAE)	10.21	12.75	8.885
Mean squared error (MSE)	169.73	342.58	157.98
Root mean squared error (RMSE)	13.03	18.51	12.57
R square (R^2)	0.889	0.780	0.897

Tabela 9: Comparação das métricas de desempenho dos algoritmos SVM, RandomForest e KNN.

Pela Tabela 10, a rede neural obteve o melhor resultado entre todos e as redes convolucionais obtiveram resultados próximos ao do SVM e KNN. Na figura 18 pode ser visto a predição feita pela rede neural profunda e o primeiro modelo da rede convolucional.

Métrica	DNN	CNN1	CNN2
Mean squared error (MSE)	127.45	161.36	196.09
Root mean squared error (RMSE)	9.355	10.651	11.29
R square (R^2)	0.918	0.887	0.863

Tabela 10: Comparação das métricas de desempenho da DNN e das CNNs.



(a) Predição utilizando modelo com rede neural profunda.

(b) Predição utilizando modelo com rede convolucional

Figura 18: Comparativo das predições entre DNN e CNN.

5.2.6 Prédio 2 - Andar 1

Para o andar 1 do prédio 2 foi utilizado o conjunto de treinamento/validação da Figura 19.

Observando a Tabela comparativa 11 o KNN se mantém consistente com uma boa acurácia, apesar de que com um score R2 menor do que nos outros prédios.

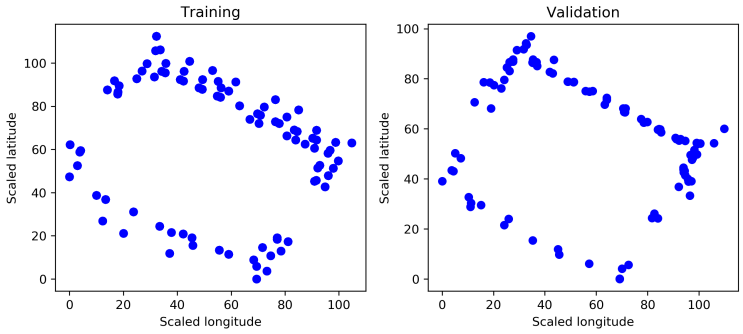


Figura 19

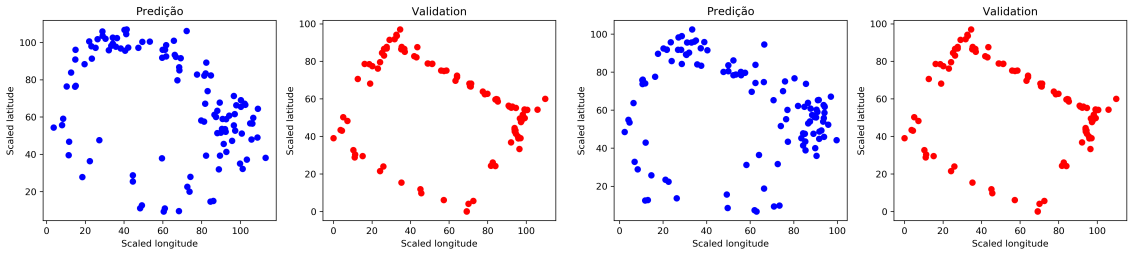
Métrica	SVM	RandomForest	KNN
Mean absolute error (MAE)	13.00	10.645	7.815
Mean squared error (MSE)	273.39	191.45	111.56
Root mean squared error (RMSE)	16.56	13.83	10.56
R square (R^2)	0.632	0.729	0.8385

Tabela 11: Comparação das métricas de desempenho dos algoritmos SVM, RandomForest e KNN.

Pela Tabela 10, as redes convolucionais desempenharam muito melhor nesse conjunto de dados do que os outros métodos. Na figura 20 pode ser visto a predição feita pela rede neural profunda e o primeiro modelo da rede convolucional.

Métrica	DNN	CNN1	CNN2
Mean squared error (MSE)	144.11	85.188	92.98
Root mean squared error (RMSE)	10.38	8.27	8.462
R square (R^2)	0.781	0.878	0.866

Tabela 12: Comparação das métricas de desempenho da DNN e das CNNs.



(a) Predição utilizando modelo com rede neural profunda.

(b) Predição utilizando modelo com rede convolucional

Figura 20: Comparativo das predições entre DNN e CNN.

6 Conclusões

Após o que foi apresentado anteriormente, pode-se concluir que para o conjunto de dados com RSSI de dispositivos *Bluetooth* foi necessário realizar alterações para que este resultasse em uma boa acurácia usando métodos de classificação. Já utilizando regressão linear, nenhuma modificação foi realizada e o resul-

tado apresentado foi satisfatório. Portanto, esta é sem dúvidas a melhor técnica a ser utilizada com o *dataset*.

Em relação ao conjunto de dados UjiIndoorLoc, o estudo feito a partir de métodos de regressão demonstrou que o KNN desempenha muito bem para o propósito de localização *indoor* com medidas de RSSI, mantendo a consistência em todos os experimentos realizados sendo superior ao SVM e ao *RandomForest*.

Em relação às redes neurais profundas e convolucionais, foram obtidos resultados tão bons quanto ao KNN e desempenhando melhor do que ele em alguns casos. Os modelos apresentados não são otimizados, podendo ser feitos mais testes de hiperparâmetros, variações de redes, podendo trazer resultados ainda melhores.

Em especial às redes convolucionais, o dataset não permitiu que no preprocessamento fosse transformado em uma imagem que representasse fielmente as posições das WLANs e portanto compromentendo o desempenho. Porém, mesmo com esse prejuízo no conjunto de dados do prédio 2 elas foram as que desempenharam melhor.

Referências

[1] A. Yassin et al., "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications,"in IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 1327-1346, Secondquarter 2017.

[2] Pu, Chuan Chin & Pu, Chuan-Hsian & Lee, Hoon-Jae. (2011). Indoor Location Tracking Using Received Signal Strength Indicator. 10.5772/10518.

[3] Chai, Song & An, Renbo & Du, Zhengzhong. (2016). An Indoor Positioning Algorithm using Bluetooth Low Energy RSSI. 10.2991/amsee-16.2016.72.

[4] Turgut, Zeynep & Gurkas Aydin, Zeynep & Sertbaş, Ahmet. (2016). Indoor Localization Techniques for Smart Building Environment. Procedia Computer Science. 83. 1176-1181. 10.1016/j.procs.2016.04.242.

[5] Qian Dong and W. Dargie, "Evaluation of the reliability of RSSI for indoor localization,"2012 International Conference on Wireless Communications in Underground and Confined Areas, Clermont Ferrand, 2012, pp. 1-6.

[6] A. H. Salamah, M. Tamazin, M. A. Sharkas and M. Khedr, "An enhanced WiFi indoor localization system based on machine learning,"2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcala de Henares, 2016, pp. 1-8.

[7] Mascharka, David & Manley, Eric. (2015). Machine Learning for Indoor Localization Using Mobile Phone-Based Sensors.

[8] S. Bozkurt, G. Elibol, S. Gunal and U. Yayan, "A comparative study on machine learning algorithms for indoor positioning,"2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA), Madrid, 2015, pp. 1-8.

[9] L. Xiao, A. Behboodi and R. Mathar, "A deep learning approach to fingerprinting indoor localization solutions,"2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, 2017, pp. 1-7.

[10] Y. Wang, Q. Ye, J. Cheng and L. Wang, "RSSI-Based Bluetooth Indoor Localization,"2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), Shenzhen, 2015, pp. 165-171.

[11] H. Hoshi, H. Ishizuka, A. Kobayashi and A. Minamikawa, "An indoor location estimation using BLE beacons considering movable obstructions,"2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU), Toyama, 2017, pp. 1-2.

[12] Chai, Song & An, Renbo & Du, Zhengzhong. (2016). An Indoor Positioning Algorithm using Bluetooth Low Energy RSSI. 10.2991/amsee-16.2016.72.

[13] UjiIndoorLoc: An indoor localization dataset, Kaggle. Disponível em: `https://www.kaggle.com/giantuji/UjiIndoorLoc/home`. Acesso em 29 de Outubro de 2018.

[14] BLE RSSI Dataset for Indoor localization, Kaggle. Disponível em: `https://www.kaggle.com/mehdimka/ble-rssi-dataset/home`. Acesso em 29 de Outubro de 2018.

[15] E. Jedari, Zheng Wu, R. Rashidzadeh and M. Saif, "Wi-Fi based indoor location positioning employing random forest classifier,"2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Banff, AB, 2015, pp. 1-5. doi: 10.1109/IPIN.2015.7346754

[16] Ke Shi, Zhenjie Ma, Rentong Zhang, Wenbiao Hu, and Hongsheng Chen, "Support Vector Regression Based Indoor Location in IEEE 802.11 Environments," Mobile Information Systems, vol. 2015, Article ID 295652, 14 pages, 2015. https://doi.org/10.1155/2015/295652.

[17] J. Torres-Sospedra et al., "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems,"2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, 2014, pp. 261-270. doi: 10.1109/IPIN.2014.7275492