1. No change to original sklearn algorithm - inicialmente testaremos a acurácia do modelo com k = 3 e training size de 70%, medidas comumente adotadas para este tipo de algoritmo. Os outros hiperparametros serão os defaults do sklearn.

Desta forma obtivemos uma acurácia de 99,36%, um resultado já bastante alto.

2. hyperparameter tuning + cross validation - nessa etapa testaremos os hiperparametros ideais e eliminaremos, através do cross validation, incertezas com relação a quais "pedaços" do dataset foram escolhidos durante o primeiro teste.

Lembrando que no primeiro teste os seguintes hiperparametros foram utilizados: {k=3, leaf_size = default = 30, weights default = 'uniform' e algorithm = default='auto'}

os seguintes parâmetros foram testados no GridSearch:

k_range = range(1, 31)

leaf_range = 2 ** np.arange(10) ---> gera o seguinte resultado: array([1, 2, 4, 8, 6,32,64, 128, 256, 512])

weight_options = ['uniform', 'distance']

algorithm_options = ['ball_tree', 'kd_tree', 'brute']

Ao final dos testes, os hiperparametros escolhidos são os seguintes: {k = 4, leaf_size = 1, weights = 'uniform', algorithm = 'ball_tree'}

e obtivemos uma acurácia de 99,21%, uma redução se comparado ao teste inicial. Seria difícil descrever exatamente o porquê deste resultado, porém, a resposta possivelmente se encontra no fato que o resultado é obtido a partir da pontuação média de validação cruzada do best_estimator, de forma que não necessariamente a acurácia abaixou, mas na realidade encontramos uma acurácia media, uma representação geral da acurácia do nosso algoritmo.

3. hyperparameter tuning + cross validation + Normalising - Esta etapa normalmente seria feita imediatametne logo após ler o dataset, com o objetivo de evitar uma escolha erronia do vizinho mais próximo, mas como estamos comparando a efetividade de etapas no processo de otimização do algoritmo somente decidi inclui-la agora. Nesta referência encontramos uma explicação sucinta de como a normalisação ajuda o algoritmo de knn:

https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

Surpreendentemente conseguimos um resultado perfeito, com 100% de acurácia. Apesar de incomum (ao ponto de ser uma indicação de erro por parte do programador) no mundo dos algoritmos de predição, este resultado faz sentido se considerarmos o dataset no qual estamos trabalhando: medidas de wi-fi em salas diferentes.

Um problema no qual deve existir uma formula que descreve perfeitamente as medidas na forma de um resultado (considerando a ausência de erros de medição), afinal uma sala possui tamanha definido e não variável.