WiFind

Gabriele Aprile Matteo Cardellini Andreea Scrob

March 2025

Contents

1 Introduction

User localization inside buildings is an important problem in many domains, such as security, home automation, and indoor navigation. One of the most widely used techniques to determine a user's position is **WiFi fingerprinting**[1], which leverages the received signal strength (RSSI[2]) from several access points and employs machine learning models to estimate position.

1.1 Problem description

The problem addressed here is to determine, as precisely as possible, the location of a user in an indoor environment using only network data, avoiding extra hardware such as GPS or external sensors.
In our case, the objective is to *identify which room of the Computer Science Department a given user is located in*, based solely on information from AlmaWiFi access points.

1.2 Proposed solutions

To tackle the problem we propose to:

- select a well-defined building area and collect data within it to **construct a dataset**;
- **choose a simple and effective machine learning model**, such as *K-Nearest Neighbors (KNN)* or *Random Forest*, to classify locations;
- **evaluate relevant metrics** and **tune the model** to achieve satisfactory performance.

2 Proposed methods

2.1 Dataset

2.1.1 Collection and preprocessing

We created a project-specific **dataset** by collecting measurements from the various *access points* in the Ercolani building (map shown in Figure 1) using a set of Python scripts. To evaluate the impact of obstacles on radio propagation, which can cause interference, measurements were taken twice for each room: once with the room empty (*Empty*) and once with the room crowded (*Crowded*).
After collection, data were filtered to keep only records related to the **AlmaWIFI** network, as shown in Table 1.

2.1.2 Exploratory data analysis

Using plots, we conducted preliminary analyses on the collected data.
In particular, we produced a **histogram** showing the total number of detections for each access point (AP) (see Figure 2). We also generated a **heatmap** (Figure 3) of WiFi signal strength that highlights the average intensity recorded for each AP across different areas (E1, E2, E3, Corridor, Garden) and the two environmental conditions (Empty, Crowded). The colours in the heatmap visually represent the received signal strength.
These plots clarified the dataset structure and revealed that some APs were not very informative (few detections), suggesting further cleaning and feature selection were needed.

[1] A localization technique based on collecting and analyzing the "fingerprints" of WiFi signals in a given area. Each location has a unique *signature* produced by the combination of signals received from multiple access points, which is then used to estimate the user's position.

[2] *Received Signal Strength Indicator* denotes the power of the signal received by a device, measured in dBm (decibels relative to 1 mW).

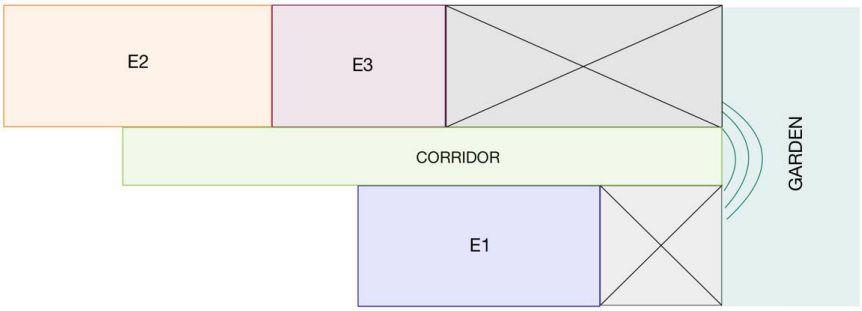

Figure 1: Map of the Ercolani building with the rooms used for data collection

oprule Room	Condition	AP1	AP2	...	AP41	AP42
E1	Empty	-76.0		...	-89.0	
E2	Empty	-45.0	-66.0	...		
E3	Empty	-64.0	-79.0	...		
Corridor	Empty	-48.0	-66.0	...		
Garden	Empty			...		-91.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
E1	Crowded	-84.0		...	-91.0	
E2	Crowded	-47.0	-64.0	...		
E3	Crowded	-66.0	-78.0	...		
Corridor	Crowded		-93.0	...		
Garden	Crowded	-78.0		...		

Table 1: Excerpt of the dataset *wifi_fingerprinting_dataset_raw.csv*

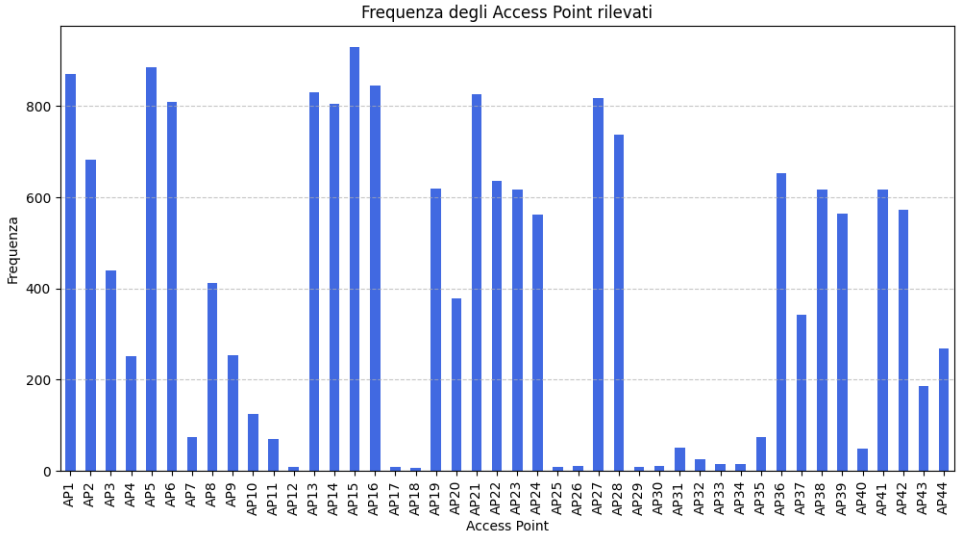

Figure 2: Histogram of total detections per Access Point (AP)

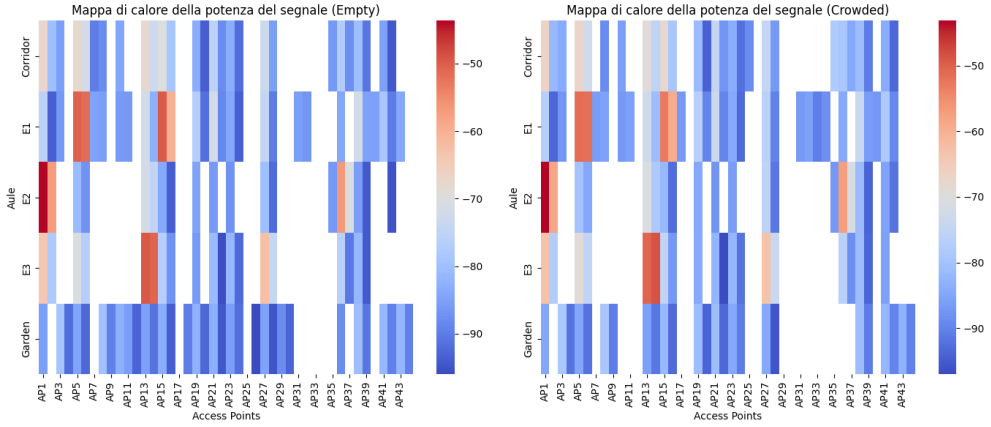

Figure 3: Heatmap of average WiFi signal strength in *Empty* and *Crowded* conditions

2.1.3 Dataset manipulations

As previously noted, some APs were detected far less frequently than others (Figure 2). To address this, we **reduced the presence of under-represented APs** by removing any AP with fewer than 400 detections using a Python script. The resulting dataset is called **cut400**.

This step also made the dataset more "tall" (fewer features relative to the number of instances), reducing the risk of high-dimensional feature space issues that can hurt models such as KNN.

We created additional variants: a normalized dataset and a binarized dataset.

In the **normalized** dataset, each RSSI value was scaled to the range [0,1] using Min–Max normalization:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

In the **binarized** dataset, values indicate only signal presence or absence: **1** for presence and **0** for absence. An excerpt is shown in Table 2.

oprule Room	Condition	AP1	AP2	...	AP41	AP42
E1	Empty	1	0	...	1	0
E2	Empty	1	1	...	0	0
E3	Empty	1	1	...	0	0
Corridor	Empty	1	1	...	0	0
Garden	Empty	0	0	...	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
E1	Crowded	1	0	...	1	0
E2	Crowded	1	1	...	0	0
E3	Crowded	1	1	...	0	0
Corridor	Crowded	0	1	...	0	0
Garden	Crowded	1	0	...	0	0

Table 2: Excerpt of the file *wifi_fingerprinting_dataset_binarized.csv*

To make *cut400* compatible with model training, missing values were replaced with **-200**, a synthetic value representing "an access point too far to be detected." This substitution is not applied to the *binarized* and *normalized* variants, which by construction contain values for all fields.

Further discussion about the use of these alternative datasets appears later.

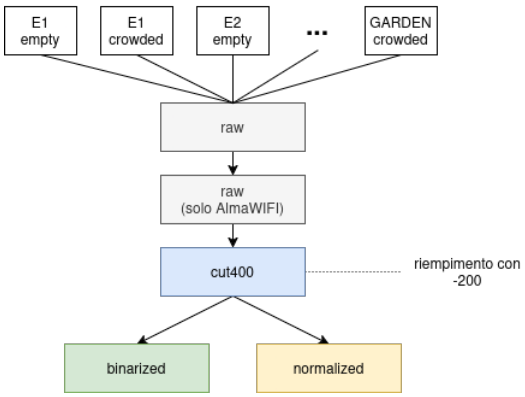

Figure 4: Dataset evolution diagram: from raw to cut400, normalized and binarized

2.2 Models

We followed a methodological approach grounded in our academic background to structure the work rigorously.

Since the task is a **classification** problem, we started with a K-Nearest Neighbors (**KNN**) implementation as a simple, well-known baseline. KNN is a supervised algorithm based on proximity: it assigns a label to a new sample based on the majority label among its **k** nearest neighbors according to a chosen distance metric. This baseline provides a reference for evaluating later, more sophisticated approaches.
We also experimented with more advanced algorithms such as Random Forest.
In the literature we found a relevant paper [1] that compares several machine learning methods for WiFi fingerprinting. It highlights the trade-offs of different approaches, for example:

- **KNN**: well-suited for moderate-sized datasets and easy to implement and tune; less recommended when the number of features becomes very large.
- **SVM**: effective in high-dimensional spaces by finding separating hyperplanes; can be sensitive to outliers and overlapping classes.
- **Random Forest**: robust and generalizes well for large datasets; as an ensemble of decision trees it reduces overfitting and yields stable predictions.

Since our datasets are moderate in size and the feature count never exceeds 44 (maximum number of detected APs), we focused on KNN and hyperparameter tuning. If KNN proved insufficient, we planned to explore other classifiers more deeply.

2.3 K-Nearest Neighbors

2.3.1 Dataset splitting

To evaluate robustness under different crowding conditions, we created three dataframes corresponding to **Empty** (room empty), **Crowded** (room crowded) and **Hybrid** (combined) scenarios.
We defined a *split_data* function that splits each subset into a **training set** (70%) and a **test set** (30%) using stratified sampling to preserve class proportions. No explicit validation set was created; validation is performed with the technique described next.

2.3.2 Model optimization

While researching KNN improvements we considered approaches from the literature such as Weighted KNN (WKNN) [2], which uses distance-based weighting to improve neighbor selection.
We chose a well-established approach: **Grid Search** to systematically find optimal hyperparameters.
extbfGrid Search: searching for optimal hyperparameters
Grid Search finds the best combination of **hyperparameters**[3] by exhaustively evaluating configurations. The process consists of:

1. defining a grid of possible values for each hyperparameter;
2. training the model on every combination in the grid;
3. evaluating each combination with a chosen metric (e.g. accuracy) and selecting the best.

The grid we explored is shown in Table 3.

oprule extbfParameter	**Values explored**
extttn_neighbors	3–19
extttweights	`['uniform', 'distance']`
extttalgorithm	`['ball_tree', 'brute']`
extttmetric	`['euclidean', 'manhattan']`

Table 3: Grid of values explored via Grid Search

Overview of optimized parameters:

- **n_neighbors**: the K value in KNN. Too small may overfit; too large may underfit. The range 3–19 was chosen as a practical compromise.
- **weights**: either *uniform* (equal vote) or *distance* (closer neighbors weighted more). Both were tested.
- **algorithm**: neighbor search method (*ball_tree*, *brute*). Given the dataset size, *ball_tree* and *brute* were appropriate choices.
- **metric**: distance metric (*euclidean*, *manhattan*). Both were evaluated.

Other hyperparameters (e.g. **p**, **leaf_size**) were left at default values as they were considered less influential.
extbfCross Validation: robust validation of hyperparameter choices
Grid Search was implemented using Scikit-Learn's *GridSearchCV*, which evaluates each hyperparameter combination with **Cross Validation**. Training data are split into K folds; each iteration trains on K-1 folds and validates on the remaining fold. We used Stratified K-Fold to preserve class proportions. The mean performance across folds provides a stable estimate for selecting the best configuration.

2.3.3 Evaluation metrics

After selecting the best hyperparameters, model performance was evaluated using:

- **Accuracy**: fraction of correctly classified instances.
- **Precision**: fraction of correct positive predictions among all positive predictions.

[3]Parameters not learned by the model but set before training.

- **Recall**: fraction of true positives correctly identified.
- **F1-score**: harmonic mean of Precision and Recall.

3 Results and discussion

After testing on different dataset variants, we found the **binarized** dataset (presence/absence) to be the most informative for our experiments compared to datasets with raw RSSI values.
This choice was motivated by suspected *overfitting* when training on precise RSSI values: with a relatively small dataset, models may learn overly specific patterns, reducing generalization. Binarization improves generalization and makes results easier to interpret.

3.1 Training

Using GridSearchCV we found the best hyperparameters for each of the three scenarios (Empty, Crowded, Hybrid), reported in Table 4.

extbfScenario	**Algorithm**	**Metric**	**Neighbors**	**Weights**	**Best Accuracy**
Empty	brute	manhattan	6	distance	0.9793
Crowded	ball_tree	manhattan	15	distance	0.9340
Hybrid	brute	euclidean	16	uniform	0.9302

Table 4: Optimal KNN configurations for the three scenarios, with corresponding accuracies

For two scenarios the *brute* algorithm was selected, reflecting that the dataset size allows exhaustive distance computation. The *manhattan* metric was chosen in two cases, indicating a tendency to benefit from Manhattan distances in this feature space. The models were trained using these parameters.

3.2 Testing

We evaluated each trained model across all combinations of training and testing scenarios; results are shown in Table 5.

extbfTrain Model	**Test Model**	Accuracy	Precision	Recall	F1-score
Empty	Empty	0.932	0.935	0.932	0.931
Empty	Crowded	0.813	0.808	0.813	0.797
Empty	Hybrid	0.912	0.912	0.912	0.909
Crowded	Empty	0.904	0.913	0.904	0.904
Crowded	Crowded	0.887	0.884	0.887	0.884
Crowded	Hybrid	0.919	0.922	0.919	0.919
Hybrid	Empty	0.959	0.962	0.959	0.959
Hybrid	Crowded	0.907	0.905	0.907	0.905
Hybrid	Hybrid	0.929	0.933	0.929	0.929

Table 5: KNN model evaluation: metrics for all training/testing combinations

From the confusion matrices in Figure 5 and the metrics in Table 5, the model generally performs **better when training and test datasets match**, with an exception: the *Crowded–Crowded* case is the worst performing, showing lower accuracy even compared to some cross-scenario evaluations.
The *Hybrid* dataset proved the most adaptable since it trains on all conditions and yields

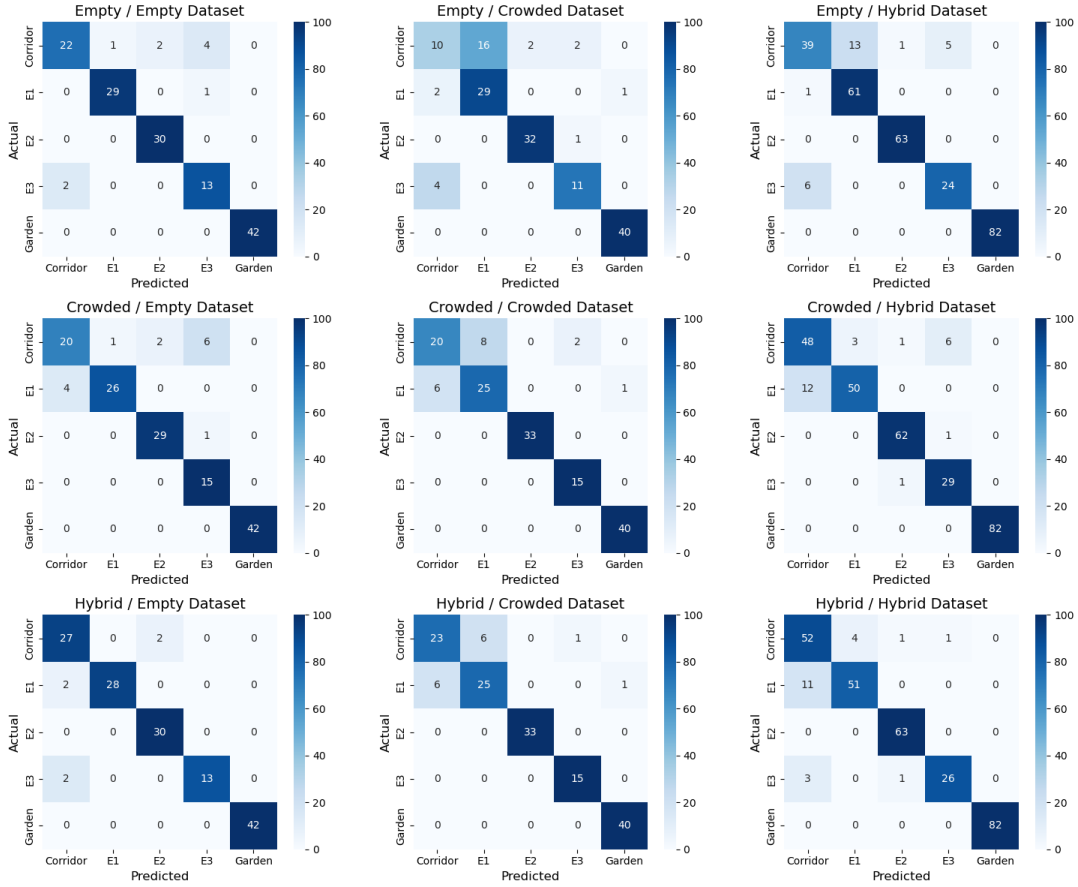

Figure 5: Confusion matrices obtained from the optimized KNN model (GridSearchCV) on the *binarized* dataset

globally better classification performance.

Overall, crowd presence does not appear to be a decisive factor for localization accuracy, although it introduces additional variability and noise.
The most frequent errors occur between **"Corridor" and "E1"**, which share the largest common area (Figure 1); overlapping router coverage likely causes confusion. The **"Garden"** area is correctly classified in most cases due to its spatial separation from other rooms.

4 Conclusion

This study analyzed the effectiveness of *WiFi fingerprinting* for indoor localization, evaluating benefits and limitations. The results **confirm the feasibility of the method** while highlighting areas for improvement.
We showed that using this technique, even with a relatively simple model such as **KNN**, provides sufficiently precise localization: it is **possible to identify a user inside the Computer Science Department using only radio signals from AlmaWIFI routers**.

4.1 Future work

The current model may not remain robust if room layouts change or if the system is deployed in a different building. For example, including additional areas or changing the building

layout could reduce performance.
An extension would be to pursue *higher localization granularity*: rather than identifying only the room, localize a more precise area inside a room.
Achieving this would require *collecting a larger dataset or adopting alternative methodologies*, such as logging exact measurement positions. A larger dataset would also allow exploring more advanced models (e.g. SVM, Random Forest) that may further improve localization performance.

References

[1] S. Shang and L. Wang, "Overview of wifi fingerprinting-based indoor positioning," *IET Communications*, vol. 16, no. 7, pp. 725–733, 2022. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cmu2.12386

[2] J.-H. Park, D. Kim, and Y.-J. Suh, "Wknn-based wi-fi fingerprinting with deep distance metric learning via siamese triplet network for indoor positioning," *Electronics*, vol. 13, no. 22, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/22/4448