WiFind

Gabriele Aprile 0001186570
Matteo Cardellini 0001186864
Andreea Scrob 0001186874

A.A. 2024/2025 – March 2025

Indice

1 Introduzione

La localizzazione degli utenti all'interno di ambienti è un problema di grande interesse in diversi ambiti, come la sicurezza, la domotica e la navigazione indoor. Una delle tecniche più usate per determinare la posizione di un utente è il **WiFi fingerprinting**[1], che sfrutta la potenza del segnale ricevuto (RSSI[2]) da più punti di accesso per inferire la posizione tramite modelli di machine learning.

1.1 Descrizione del problema

Il problema in analisi consiste nel determinare con la massima precisione possibile la posizione di un utente in un dato ambiente indoor utilizzando solo i dati di rete, evitando l'uso di hardware aggiuntivo come GPS o sensori esterni.

Nel nostro caso, il problema è *determinare in quale stanza del Dipartimento di Informatica si trovi un dato utente*, basandoci sulle informazioni degli access point della rete universitaria AlmaWiFi.

1.2 Soluzioni proposte

Per affrontare il problema, si propone di:

- scegliere una struttura ben definita e raccogliere i dati al suo interno, in modo da poter **creare un dataset**;
- **selezionare un modello di machine learning** semplice ed efficace, come *K-Nearest Neighbors (KNN)* o *Random Forest*, per la classificazione della posizione;
- **valutare le metriche** e **ottimizzare il modello** in modo da ottenere risultati soddisfacenti.

2 Metodi proposti

2.1 Dataset

2.1.1 Raccolta e preprocessing

Si è deciso di creare un **dataset ad-hoc** per il progetto, raccogliendo i dati relativi ai vari *access point* dell'edificio Ercolani (la cui cartina è visibile in Figura 1), attraverso una serie di script Python. Per valutare l'impatto degli ostacoli sulle onde radio, un fattore che potrebbe causare interferenze, la rilevazione è stata effettuata due volte per ogni stanza: una con la stanza vuota (*Empty*) e una con la stanza affollata (*Crowded*).
Al termine della raccolta, i dati sono stati filtrati conservando esclusivamente quelli relativi alla rete **AlmaWIFI**, come mostrato in Tabella 1.

2.1.2 Analisi Esplorativa dei dati

Grazie al supporto di grafici, sono state condotte analisi preliminari sui dati raccolti.
In particolare, è stato realizzato un **istogramma** per esaminare il numero totale di segnali rilevati da ciascun access point (AP) (vedi Figura 2). È stata inoltre generata una **heatmap** (Figura 3) della potenza del segnale Wi-Fi, che evidenzia l'intensità media rilevata per

[1] tecnica di localizzazione basata sulla raccolta e analisi delle "impronte digitali" dei segnali WiFi in una determinata area. Ogni posizione ha una *firma* unica data dalla combinazione dei segnali ricevuti dai vari access point, che viene utilizzata per stimare la posizione dell'utente.

[2] *Received Signal Strength Indicator* indica la potenza del segnale ricevuto da un dispositivo, misurata in dBm (decibel rispetto a 1 mW).

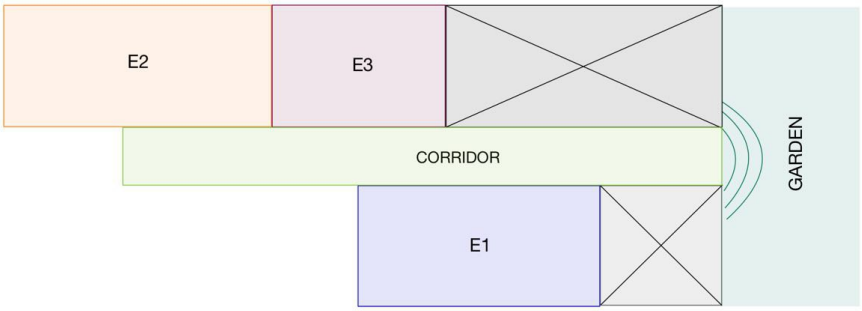

Figura 1: Mappa dell'edificio Ercolani con le stanze usate per la raccolta dati

Aula	Situazione	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		...		

Tabella 1: Estratto del dataset *wifi_fingerprinting_dataset_raw.csv*

ogni AP all'interno di diverse aree (E1, E2, E3, Corridoio, Giardino), nelle due differenti condizioni ambientali (Empty, Crowded). I colori della mappa riflettono visivamente la forza del segnale ricevuto.
Questi grafici hanno permesso una comprensione più chiara della struttura del dataset, evidenziando come alcuni access point risultassero poco significativi a causa del numero esiguo di rilevazioni associate, suggerendo la necessità di effettuare ulteriori operazioni di pulizia e selezione dei dati.

2.1.3 Manipolazioni del dataset

Come detto in precedenza, si è notato come alcuni access point venissero rilevati meno frequentemente rispetto ad altri (Figura 2). Per questo motivo, si è scelto di **ridurre la presenza di AP** poco rappresentativi nel dataset. A tale scopo, è stato utilizzato uno script Python che ha rimosso tutti gli AP con meno di 400 rilevazioni, mantenendo inalterati gli altri. Il dataset ottenuto viene indicato con il nome **cut400**.
Questa operazione ha avuto anche l'effetto positivo di rendere il dataset più "verticale", ovvero con un numero più contenuto di feature (access point) rispetto al numero totale di istanze. In questo modo, si è ridotto il rischio di sovradimensionamento dello spazio delle feature, che potrebbe altrimenti compromettere l'efficacia di modelli come KNN.

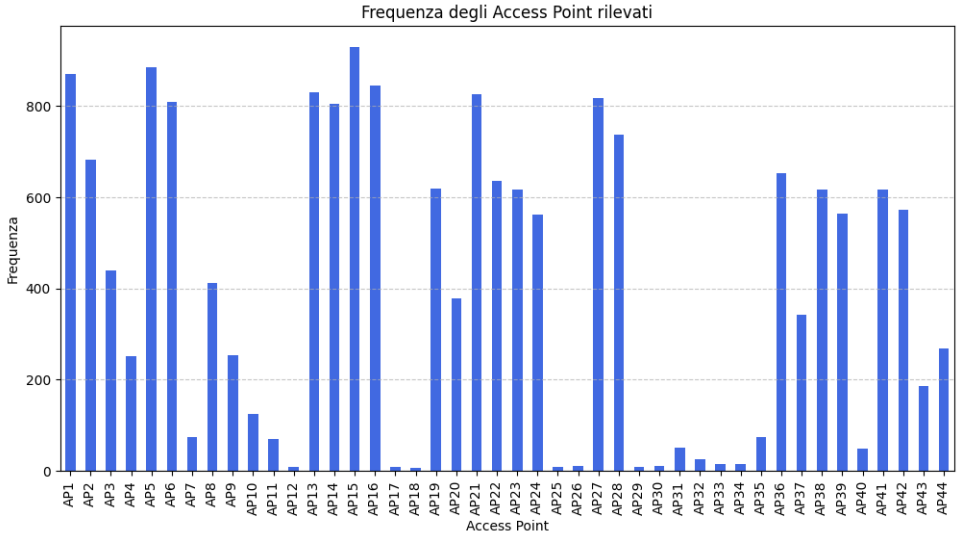

Figura 2: Istogramma del numero totale di rilevazioni per ciascun Access Point (AP)

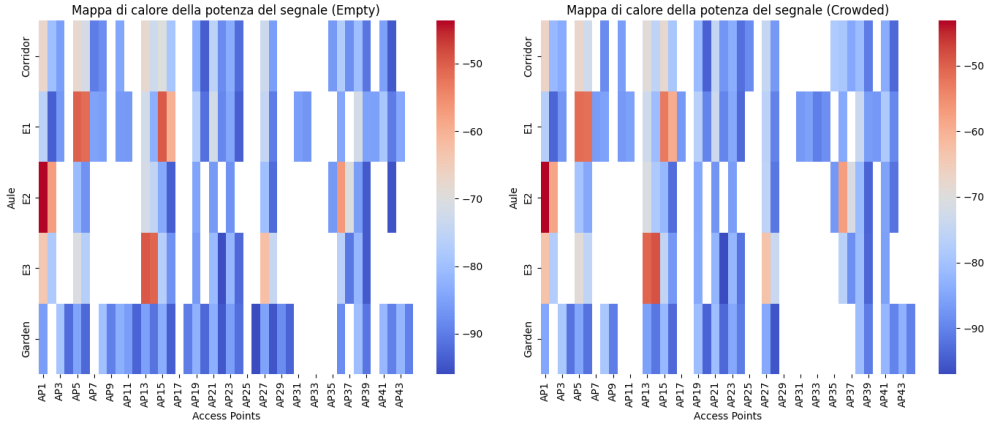

Figura 3: Heatmap della potenza media del segnale Wi-Fi nelle condizioni *Empty* e *Crowded*

Sono stati generati ulteriori dataset, tra cui uno con *valori normalizzati* e uno con *valori binarizzati*.
Nel primo caso, denominato **normalized**, ogni valore di potenza RSSI è stato scalato in un intervallo compreso tra 0 e 1, con una precisione di 16 cifre decimali. Per la normalizzazione è stato adottato il metodo Min-Max, secondo la formula:

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

Nel dataset binarizzato, chiamato **binarized**, l'obiettivo è invece quello di rappresentare la semplice presenza o assenza del segnale, piuttosto che la sua intensità. A tal fine, i valori RSSI sono stati convertiti in binari: in presenza di segnale è stato assegnato il valore **1**, in assenza il valore **0**. Il risultato di questa trasformazione è mostrato in Tabella 2.

È importante sottolineare che, per rendere *cut400* compatibile con l'addestramento dei modelli, i campi privi di rilevazioni sono stati sostituiti con il valore **-200**, un valore fittizio che rappresenta "un access point troppo distante per essere rilevato".

Aula	Situazione	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

Tabella 2: Estratto del file *wifi_fingerprinting_dataset_binarized.csv*

Questo ragionamento non viene ovviamente applicato per *binarized* e *normalized*, che per sua natura hanno tutti i campi contenenti un valore.

Ulteriori motivazioni e considerazioni riguardanti l'utilizzo di questi dataset alternativi verranno approfondite successivamente.

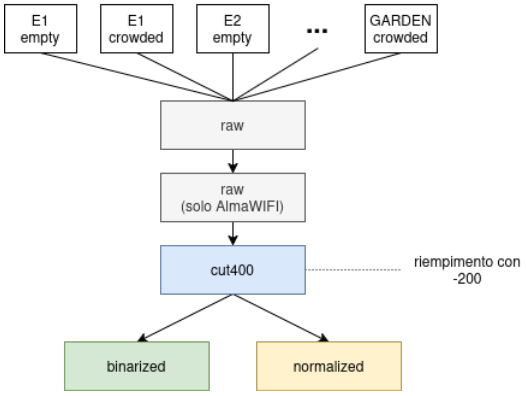

Figura 4: Schema dell'evoluzione del dataset: da raw a cut400, normalized e binarized

2.2 Modelli

Sin dall'inizio, abbiamo adottato un approccio metodologico basato sulle conoscenze acquisite nel nostro percorso accademico, al fine di strutturare e avviare il lavoro in modo rigoroso.

Dato il problema di **classificazione** da risolvere, come primo passo, abbiamo scelto di procedere ad una prima implementazione di **K-Nearest Neighbors** (**KNN**), un modello di machine learning noto per la sua semplicità ed efficacia, frequentemente incontrato e studiato nel corso della nostra formazione. Il **KNN** è un algoritmo supervisionato basato sul principio della vicinanza: classifica un nuovo dato assegnandogli l'etichetta predominante tra i suoi **k** vicini più prossimi, misurati tramite diverse metriche. La scelta di questo modello iniziale ci ha permesso di stabilire un punto di riferimento per valutare le prestazioni e l'affidabilità dei successivi approcci sperimentati.
L'analisi è stata approfondita tramite l'uso di algoritmi più raffinati, come ad esempio il Random Forest.

Dopo diversi tentativi e ricerche, è stata trovata una pubblicazione [1] che affronta in modo specifico il problema della localizzazione indoor tramite WiFi fingerprinting.
Nel paper viene condotta un'analisi comparativa tra diversi algoritmi di Machine Learning, evidenziandone i punti di forza e le limitazioni in relazione a questo specifico contesto applicativo.
Vengono citati:

- **K-Nearest Neighbour (KNN)**: particolarmente adatto a dataset di dimensioni moderate, e relativamente semplice da implementare e regolare. Tuttavia non è consigliato per dataset con un elevato numero di feature, poiché la sua efficienza può risentirne.
- **Support Vector Machine (SVM)**: efficace su dataset ad alta dimensionalità, grazie alla sua capacità di trovare iperpiani di separazione ottimali tra le classi.Ciononostante, può risultare meno performante in presenza di numerosi outlier o quando le classi sono fortemente sovrapposte nello spazio delle feature, rendendo difficile la separazione dei cluster.
- **Random Forest**: particolarmente indicato per dataset di grandi dimensioni, grazie alla sua robustezza e capacità di generalizzazione. Essendo un metodo basato su un insieme di alberi decisionali, riesce a ridurre il rischio di overfitting e a garantire una maggiore stabilità nelle predizioni.

Considerando i nostri dataset di dimensioni non eccessive e di dimensionalità mai superiori a 44 (massimo numero di AP individuati), si è deciso di procedere utilizzando il modello *KNN*, puntando ad un miglioramento tramite *tuning* di parametri.
Qualora il KNN non si fosse rivelato sufficientemente performante, avremmo proceduto con un'analisi più approfondita di altri modelli di classificazione, al fine di individuare soluzioni più efficaci per il problema affrontato.

2.3 K-Nearest Neighbors

2.3.1 Divisione dataset

Al fine di verificare la robustezza del modello nelle varie possibili situazioni di affollamento delle stanze, il dataset è stato manipolato per ottenere tre dataframe diversi, corrispondenti alle situazioni "**Empty**" (aula vuota), "**Crowded**" (aula affollata) e "**Hybrid**" (situazione mista, ovvero l'unione dei due dataframe precedenti).
Per gestire correttamente la fase di addestramento e valutazione, si è definita la funzione *split_data*, che divide ogni sottoinsieme in un **training set** (70% dei dati) e un **test set** (30% dei dati). Questa suddivisione avviene in modo **stratificato**, ovvero mantenendo la distribuzione originale delle classi.
Il *validation set* non è stato separato esplicitamente in questa fase, ma il suo utilizzo è garantito tramite una tecnica descritta nel paragrafo seguente.

2.3.2 Ottimizzazione del Modello

Durante lo studio per migliorare le prestazioni del modello KNN, ci siamo imbattuti in diversi approcci affrontati dalla letteratura del caso, come ad esempio quello presentato in [2]: il metodo Weighted k-Nearest Neighbors (WKNN) permette di utilizzare metriche di distanza più sofisticate e migliorare la selezione dei vicini.
Tuttavia, abbiamo scelto un approccio più tradizionale e consolidato: **Grid Search**, che consente una ricerca sistematica degli iperparametri ottimali.

Grid Search: ricerca degli iperparametri ottimali

La *Grid Search* è una tecnica utilizzata per individuare la combinazione ottimale di **iperparametri**[3] di un algoritmo di machine learning. Nel nostro caso, è stata impiegata per ottimizzare le prestazioni del modello KNN, esplorando sistematicamente diverse configurazioni degli iperparametri.
Il processo si articola in tre fasi principali:

1. definizione di una griglia di valori possibili per ciascun iperparametro del modello;
2. addestramento del modello su tutte le combinazioni possibili della griglia;
3. valutazione delle prestazioni di ogni combinazione tramite una metrica predefinita (es. accuratezza), e selezione della migliore configurazione.

Nel nostro progetto, la griglia dei valori esplorata è riportata nella Tabella 3.

Parametro	**Valori esplorati**
`n_neighbors`	3–19
`weights`	`['uniform', 'distance']`
`algorithm`	`['ball_tree', 'brute']`
`metric`	`['euclidean', 'manhattan']`

Tabella 3: Griglia dei valori esplorati tramite Grid Search

Di seguito una panoramica dei parametri ottimizzati:

- **n_neighbors** – rappresenta il valore di K in KNN, ovvero il numero di vicini da considerare per effettuare la classificazione.
  - Un valore di K troppo basso può portare a *overfitting*, mentre valori troppo alti rischiano di causare *underfitting*.
  - Valutazione: Il range 3–19 è stato ritenuto un buon compromesso per bilanciare queste problematiche.
- **weights** – specifica l'importanza assegnata ai vicini nel processo decisionale.
  - *uniform*: tutti i vicini contribuiscono in egual misura.
  - *distance*: i vicini più prossimi hanno un peso maggiore.
  - Valutazione: Considerando la natura dei dati, l'opzione *uniform* potrebbe essere sufficiente, ma si è lasciata comunque la scelta aperta.
- **algorithm** – metodo utilizzato per calcolare i vicini più prossimi.
  - *ball_tree*: adatto a dataset di dimensioni medio-grandi e con molte feature sparse.
  - *brute*: approccio esaustivo che calcola la distanza tra tutte le coppie di punti; adatto a dataset piccoli.
  - *kd_tree*: efficiente con dataset bilanciati e a bassa dimensionalità.
  - Valutazione: Considerate le dimensioni contenute del dataset, *ball_tree* e *brute* sono risultati i più adatti.
- **metric** – criterio di distanza utilizzato per calcolare la somiglianza tra i punti.
  - *euclidean*: la distanza euclidea classica tra punti nello spazio.
  - *manhattan*: somma delle distanze assolute per ogni dimensione; più robusta in spazi ad alta dimensionalità.

[3] Parametri che non vengono appresi dal modello, ma devono essere definiti a priori.

- *minkowski*: generalizzazione delle due precedenti.
- Valutazione: Dato il numero limitato di feature, si è scelto di testare soltanto le metriche *euclidean* e *manhattan*.

Altri iperparametri disponibili per il modello KNN, ma ritenuti meno influenti nel nostro contesto, sono stati lasciati ai valori di default:

- **p** – esponente utilizzato nella metrica Minkowski.
- **leaf_size** – numero massimo di punti che possono essere contenuti in una foglia degli alberi (*ball_tree* e *kd_tree*).

Cross Validation: validazione robusta delle scelte sui parametri

La Grid Search è implementata nel codice tramite la funzione *GridSearchCV* del pacchetto Scikit-Learn. Questa funzione consente di individuare la combinazione ottimale di iperparametri attraverso una valutazione basata su *validation test*. In particolare, sfrutta la tecnica della **Cross Validation**, suddividendo il dataset di addestramento in K sottoinsiemi (*fold*), eseguendo il training del modello K volte:

- Ad ogni iterazione, K-1 folds vengono usati per l'addestramento e il fold rimanente è considerato come validation set.
- Alla fine, si calcola la media delle prestazioni, ottenendo una valutazione più stabile e affidabile.

Abbiamo scelto di utilizzare la *Stratified K-Fold Cross Validation*, in quanto garantisce che ogni fold mantenga le stesse proporzioni delle classi presenti nel dataset originale. Questo è particolarmente importante quando le classi sono sbilanciate, poiché assicura che la distribuzione dei dati sia coerente in tutte le iterazioni.

L'integrazione di Grid Search con Cross Validation ha permesso di **ottimizzare il modello in modo rigoroso**, migliorandone l'affidabilità e la capacità di generalizzare su nuovi dati.

2.3.3 Metriche

Dopo aver trovato i migliori parametri, il modello viene testato e valutato su diverse metriche:

- **Accuracy**: misura la quantità di elementi correttamente classificati (veri positivi e veri negativi) sul totale delle previsioni.
- **Precision**: misura la quantità di elementi correttamente classificati come positivi (veri positivi) sul totale di tutti i positivi (veri positivi e falsi positivi).
- **Recall**: misura quanto bene il modello riesce a trovare tutti i veri positivi (sull'insieme di veri positivi e falsi negativi).
- **F1-score**: è la media armonica tra Precision e Recall.

3 Risultati e Commenti

Dopo aver testato il sistema sui diversi dataset, abbiamo ritenuto più significativo utilizzare il **dataset *binarized***, in cui i segnali sono rappresentati in termini di presenza o assenza (1/0), anziché attraverso i valori esatti di intensità (RSSI).

Questa scelta è stata motivata dal sospetto di *overfitting* riscontrato nei modelli addestrati con i dataset alternativi contenenti valori RSSI. La nostra ipotesi è che fornire al sistema

valori di intensità troppo precisi, a fronte di un dataset relativamente ridotto, induca il modello a cogliere pattern eccessivamente specifici, compromettendone la capacità di generalizzazione su dati nuovi.

La generalizzazione introdotta dal formato binario ha inoltre reso i test qualitativamente più significativi, evidenziando con maggiore chiarezza la complessità del problema e le differenze tra le classi.

3.1 Training

Utilizzando GridSearchCV, abbiamo trovato gli iperparametri ideali per ognuno dei tre scenari Empty, Crowded e Hybrid, ottenendo i risultati riportati nella Tabella 4.

Scenario	**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

Tabella 4: Configurazioni ottimali del modello KNN per i tre scenari, con relative accuratezze

È interessante notare come per ben due scenari l'algoritmo usato sia di tipo *brute force*, a dimostrazione del fatto che le dimensioni relativamente contenute del dataset consentono il calcolo delle distanze tra tutte le coppie di punti. Per quanto riguarda le metriche, in due casi su tre è stata scelta la *manhattan*, attestando una tendenza del dataset a presentarsi come "largo", ovvero con un numero considerevole di features.
Si è proceduto ad addestrare il modello usando gli iperparametri trovati.

3.2 Testing

Per testare in modo più rigoroso il modello, si è deciso di valutarlo su tutte le possibili combinazioni ambientali, come riportato in Tabella 5. Ad ogni combinazione sono stati associati i relativi valori delle metriche precedentemente discussi.

Train 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

Tabella 5: Valutazione del modello KNN: metriche per tutte le combinazioni di training e test

Dall'analisi delle matrici di confusione riportate in Figura 5 e dai dati in Tabella 5, emerge che il modello, nel caso di situazioni *Empty* e *Crowded*, presenta le **prestazioni migliori quando il dataset di addestramento e quello di test coincidono**, mentre è tendenzialmente meno performante quando train e test avvengono su scenari differenti.
Il dataset ***Hybrid* sembra offrire la maggiore capacità di adattamento** in quanto il modello è allenato su tutte le situazioni possibili, permettendo quindi una classificazione

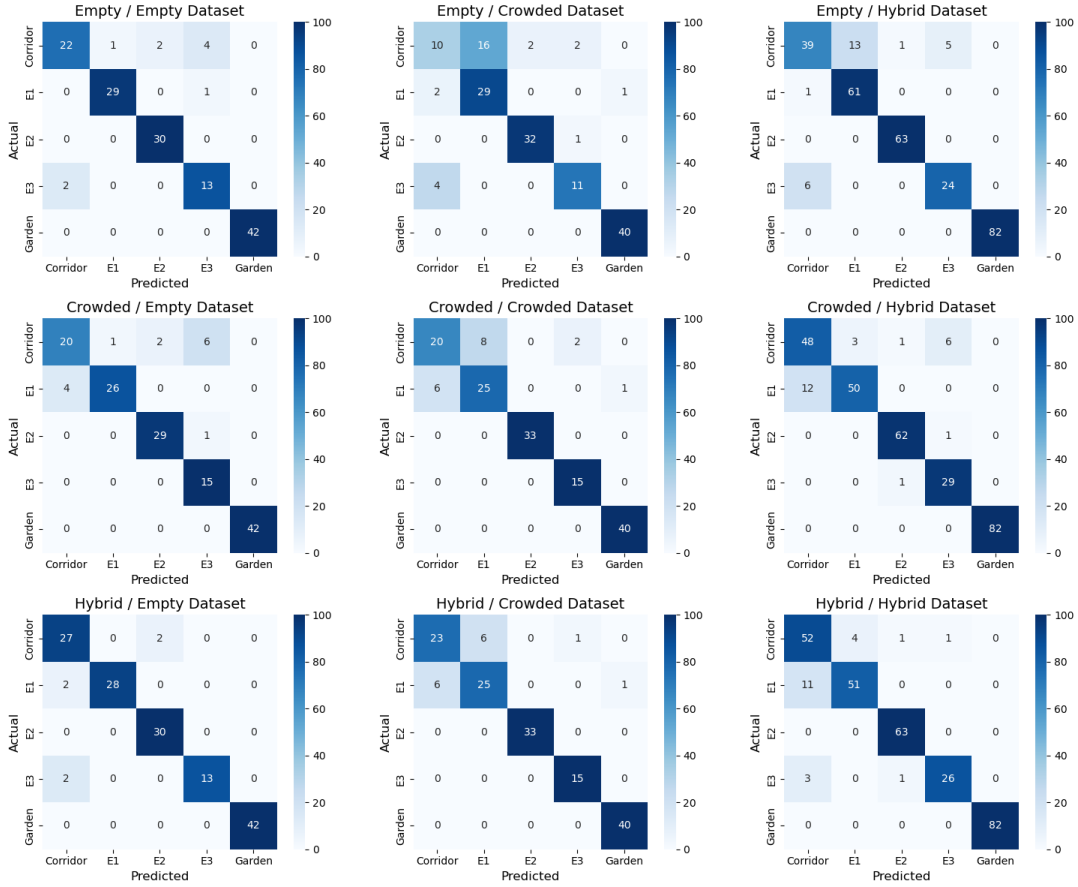

Figura 5: Matrici di confusione ottenute dal modello KNN ottimizzato (GridSearchCV) su dataset *Binarized*

globalmente più accurata.

Considerati i buoni livelli di accuratezza raggiunti per tutte le configurazioni, si può dedurre inoltre che la **presenza di folla** all'interno delle aule **non sia un fattore particolarmente rilevante** ai fini della localizzazione.

Gli **errori** più frequenti si verificano tra **"Corridor" ed "E1"**: sono due ambienti aventi la superficie in comune più estesa (Figura 1). Probabilmente, a causa della posizione dei rispettivi router, i segnali Wi-Fi si sovrappongono creando confusione nella previsione e aumentando la probabilità di errori di classificazione.
"Garden" viene **classificato correttamente** nella maggior parte dei casi: anche in questo caso la motivazione deriva dalla sua posizione all'interno dello spazio considerato. Essendo separato distintamente dalle altre stanze considerate, il modello ha una maggiore facilità nel riconoscerlo.

4 Conclusione

In questo studio abbiamo analizzato l'efficacia del *WiFi fingerprinting* per la localizzazione, valutando i vantaggi e le sfide di questo approccio. I risultati ottenuti **confermano la fattibilità del metodo**, pur evidenziando alcune limitazioni e possibili miglioramenti.

Il nostro studio ha dimostrato che l'uso di questa tecnica, anche con l'applicazione di un modello relativamente semplice come **KNN**, consente una localizzazione sufficientemente precisa: i risultati evidenziano che è **possibile individuare un utente all'interno del Dipartimento di Informatica utilizzando esclusivamente le onde radio emesse dai router AlmaWIFI**.

4.1 Sviluppi futuri

Il modello sviluppato potrebbe non essere sufficientemente robusto nel caso di *modifiche alle stanze considerate*. Ad esempio, se venissero incluse le due aree in grigio di Figura 1 o, addirittura, se fosse necessario cambiare edificio, è possibile che il sistema implementato con questo approccio non fornisca più risultati soddisfacenti.

Una possibile espansione del lavoro potrebbe essere orientata verso una *maggiore granularità nella localizzazione*. Invece di limitarsi a individuare la singola stanza, potrebbe essere interessante poter localizzare un'area più precisa all'interno della stanza stessa.

Per affrontare questo nuovo obiettivo, sarebbe necessario *raccogliere un dataset più ampio o adottare metodologie alternative*, come ad esempio registrare la posizione esatta in cui vengono rilevati i segnali.

Questa espansione del dataset potrebbe anche rappresentare un'opportunità per esplorare e *risolvere il problema utilizzando modelli più avanzati*, come SVM e Random Forest, che potrebbero contribuire a migliorare ulteriormente le prestazioni del sistema di localizzazione.

Riferimenti bibliografici

[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