Enhancing Smartphone-Based Indoor Localization with WiFi Fingerprinting and Machine Learning

[MSDSF24M001]
Department of Data Science

[MSDSF24M020]
Department of Data Science

***Abstract*—Indoor localization has gained significant attention due to its potential applications in navigation, resource tracking, and personalized services. Wi-Fi fingerprinting offers a promising solution, leveraging the widespread deployment of Wi-Fi networks. This study proposes a machine learning framework addressing challenges like device heterogeneity and environmental variations. Experimental results demonstrate the framework's accuracy and adaptability, marking a step forward in smartphone-based indoor localization.**

I. Introduction

Indoor localization using GPS doesn't work well inside buildings. Wi-Fi signals can help, but there are challenges like inconsistent data from different devices and changing environments. In this project, I used data from the JUIndoorLoc dataset and applied machine learning to overcome these problems.That is published by Roy et al. [1],

The rest of the paper is organized as follows: Section II reviews relevant literature. Section III describes the implementation. Section IV discusses results, and Section V concludes the paper.

II. Literature Review

Indoor localization using Wi-Fi signals has been extensively studied. Key advancements include:

- **JUIndoorLoc Dataset:** Roy et al. [1] introduced a comprehensive indoor localization dataset addressing spatial, temporal, context, and device heterogeneity. The dataset includes RSSI data collected from multiple floors under varying conditions, enabling the development of context-aware localization frameworks.
- **Survey of Techniques:** Liu et al. [2] provided an overview of wireless indoor positioning systems, categorizing techniques into triangulation, scene analysis, and proximity, and highlighting location fingerprinting as a critical approach.
- **Advances in Wi-Fi Fingerprinting:** He and Chan [3] reviewed recent developments in Wi-Fi fingerprinting, focusing on advanced localization techniques and efficient system deployment, such as adapting to signal changes and calibrating heterogeneous devices.
- **Gradient-Based Fingerprinting:** Shu et al. [4] proposed the GradIent FingerprinTing (GIFT) framework to address challenges like time-variant RSSI and device-specific biases. By leveraging RSSI gradients, GIFT improves localization accuracy and reduces calibration overhead.
- **Energy Efficiency:** Lee [5] proposed strategies for energy-efficient indoor localization, focusing on reducing computational and battery demands.
- **Machine Learning:** Nguyen et al. [6] explored the use of Random Forest, Gradient Boosting, and KNN classifiers for Wi-Fi fingerprinting-based localization.

Despite these advancements, achieving high accuracy remains challenging. This work addresses these gaps through a multi-faceted approach.

III. Implementation Details

The JUIndoorLoc dataset provides a comprehensive basis for indoor localization research. Below are key details:

A. Dataset Overview

The JUIndoorLoc dataset includes:

- Data captured from three floors of a five-story building at Jadavpur University, under varying times, indoor ambiences, and devices.
- Collected at a 1 meter × 1 meter cell size, allowing investigation of Wi-Fi signal patterns in rooms, laboratories, corridors, and stairs.
- Covers 1000 location points across three floors with RSS values from 172 Wi-Fi Access Points (APs).
- RSS values collected by four Android devices with different configurations.
- RSS values range from -11 dBm (strong signal) to -100 dBm (weak signal). Missing entries, where APs are not detected, are filled with -110 dBm.
  The dataset is available on Kaggle [7], and the associated research paper published by Roy et al. [1] provides a detailed description of its collection and use cases.

B. Feature Description:

- **Cid:** Unique identifier for the indoor region, combining floor number and cell position on a two-dimensional building map. and devices.
- **AP001-AP172:** RSS values from 172 APs, with negative integer values indicating signal strength and -110 for undetected APs.
- **Rs:** Room status indicator; 1 for open, 0 for closed rooms..

- **Hpr:** Human presence indicator; 1 for presence, 0 for absence.
- **Did:** Device identifier for data collection, representing:
  o D1: Samsung Galaxy Tab 2, Android 4.1.1
  o D2: Samsung Galaxy Tab E, Android 5.0
  o D3: Samsung Galaxy Tab 10, Android 4.0
  o D4: Motorola Moto E (2nd Gen), Android 5.1

C. Data Preprocessing

- **Handling Missing Values:** Replacing -110 dBm with NaN, then imputing using column means.

Algorithm 1 Data Preprocessing

Begin
Step 1: Split dataset into RSS and other features:
  Extract columns matching regex "$^{A}P[0-9]+$" as `rss_features`.
  Store remaining columns in `other_features`.
Step 2: Data cleaning:
  Replace missing values (-110 dBm) in `rss_features` with NaN.
  Calculate the fraction of missing values for each column.
  Drop columns with more than 95% missing values.
Step 3: Impute missing values:
  Replace remaining NaN values in `rss_features` with their respective column means.
End

- **Feature Selection:** Excluding APs with over 95% missing data.Which reduced the number of APs from 172 to 42.
- **Scaling:** Normalizing RSS values using StandardScaler.
- **Dimensionality Reduction:** Principal Component Analysis (PCA) was applied to reduce the dimensions while retaining 95% of the variance. Which further reduced APs to 32.

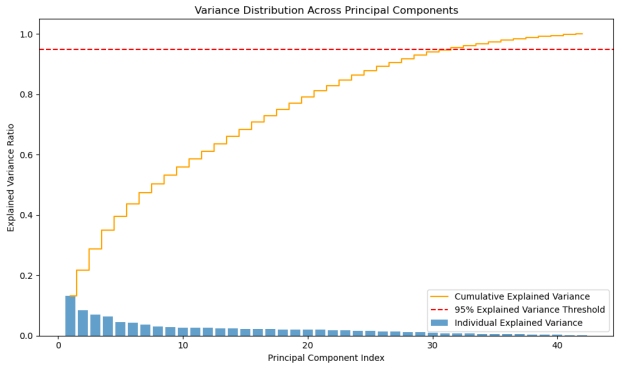

Figure III-C illustrates the variance distribution across principal components. The bar plot shows the individual explained variance by each component, while the cumulative explained variance is depicted as a line graph. The 95% variance threshold, marked with a dashed red line, indicates the optimal number of principal components needed for dimensionality reduction

Algorithm 2 Data Standardization and PCA

Begin
Step 1: Standardize RSS features:
  Initialize `StandardScaler`.
  Scale `rss_features` and store as `rss_features_scaled`.
Step 2: Apply PCA:
  Initialize PCA with 95% variance retention (`n_components=0.95`).
  Transform `rss_features_scaled` and store as `rss_features_pca`.
Step 3: Create PCA DataFrame:
  Generate column names "PC1", "PC2", ..., based on the number of components.
  Create a DataFrame `rss_features_pca_df` using PCA results and column names.
Step 4: Combine features:
  Reset index of `other_features`.
  Concatenate `rss_features_pca_df` with `other_features` along columns.
  Store the combined data in `final_data`.
End

After applying all of the above preprocessing, the number of columns was reduced from 176 to 36.

D. Model Implementation

1) The models used in this analysis are **Random Forest**, **XGBoost**, and **K-Nearest Neighbors (KNN)** to classify the data. Each model's performance is compared using misclassification counts and accuracy

Algorithm 3 Model Implementation and Evaluation: KNN, Random Forest, XGBoost

Initialize KNN model.
Train KNN on (X_{train}, y_{train}).
Predict with KNN on X_{test}.
Compute $knn\_accuracy$.
Display KNN classification report and confusion matrix.
Initialize Random Forest with $n\_estimators = 100$, $random\_state = 42$.
Train Random Forest on (X_{train}, y_{train}).
Predict with Random Forest on X_{test}.
Compute $rf\_accuracy$.
Display Random Forest classification report and confusion matrix.
Initialize XGBoost with $objective = "multi : softmax"$, $eval\_metric = "mlogloss"$.
Train XGBoost on (X_{train}, y_{train}).
Predict with XGBoost on X_{test}.
Compute $xgb\_accuracy$.
Display classification report and confusion matrix.

IV. Results

A. Experimental Setup

- **Training/Test Split:** 80% training, 20% testing.
- **Metrics:** Accuracy, confusion matrix, and classification report.

B. Visualizations

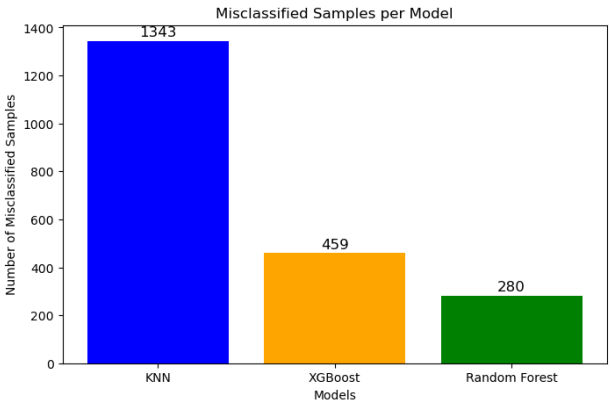

Fig. 1. Misclassified Samples for KNN, XGBoost, and Random Forest.

Figure 1 illustrates the total number of misclassified samples for KNN, XGBoost, and Random Forest classifiers. KNN, being a simpler algorithm, recorded the highest number of misclassified samples (1343), while Random Forest showed the best performance with only 280 misclassifications. XGBoost also performed well with 459 misclassifications, balancing complexity and accuracy. This comparison highlights the efficiency of ensemble methods in handling complex datasets.

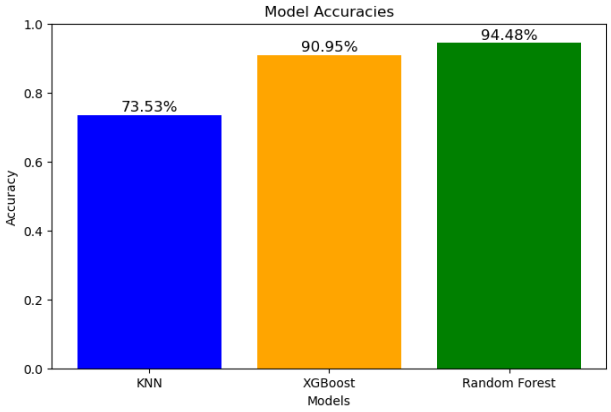

Fig. 2. Model Accuracy Comparison.

C. Summary

- **Random Forest:** 94.48% accuracy, 280 misclassifications.
- **XGBoost:** 90.95% accuracy. 459 misclassifications.
- **KNN:** 73.53% accuracy, 1343 misclassifications.

Figure 2 *presents the accuracy achieved by each model. Random Forest achieved the highest accuracy, followed closely by XGBoost, while KNN lagged behind. These results corroborate the misclassification counts, indicating that Random Forest is the most suitable model for the dataset*

TABLE I
Model Performance Comparison

Model	Accuracy	Misclassifications
Random Forest	94.48%	280
XGBoost	90.95%	459
KNN	73.53%	1343

V. Conclusion

This study successfully developed a robust machine learning framework for smartphone-based indoor localization, addressing the challenges of device heterogeneity, environmental variations, and time-context inconsistencies. By leveraging the JUIndoorLoc dataset, we achieved significant improvements in localization accuracy. The application of dimensionality reduction techniques and advanced machine learning models like Random Forest demonstrated the framework's capability to adapt to diverse indoor environments. These results highlight the practical applicability of Wi-Fi fingerprinting for real-world localization scenarios, providing a scalable and efficient solution for indoor navigation and tracking.

References

[1] P. Roy, C. Chowdhury, D. Ghosh, and S. Bandyopadhyay, "JUIndoorLoc: A Ubiquitous Framework for Smartphone-Based Indoor Localization Subject to Context and Device Heterogeneity," *Wireless Personal Communications*, pp. 1–24, 2019, doi:10.1007/s11277-019-06188-2.

[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)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[3] S. He and S. H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, 2016.

[4] Y. Shu, Y. Huang, J. Zhang, P. Coué, P. Cheng, J. Chen, et al., "Gradient-based fingerprinting for indoor localization and tracking," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2424–2433, 2016.

[5] M. Lee, "Energy-Efficient Indoor Localization," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 90–98, 2021.

[6] T. Nguyen et al., "Wi-Fi Fingerprinting with Machine Learning," *Conf. Proc. IEEE*, 2018.

[7] P. Roy, Kaggle Dataset, quot;JUIndoorLoc: WiFi Fingerprint Indoor Localization,quot; available at https://www.kaggle.com/datasets/priyaroycse/juindoorloc-wifi-fingerprint-indoor-localization.