**Indoor localization and Wi-Fi Fingerprinting**

DSAI 3201 Machine Learning: Project Part 1

Indoor localization and Wi-Fi Fingerprinting Using Machine Learning and Deep Learning

Joy Anne P. Dela Cruz[[1]](#footnote-1)\*

Student, University of Doha for Science and Technology, [60301959@udst.edu.qa](mailto:60301959@udst.edu.qa)

Asma Maged Alwan[[2]](#footnote-2)\*

Student, University of Doha for Science and Technology, 60106908@udst.edu.qa

Indoor localization is a widely used system for determining the position of devices indoors. As part of the University of Doha for Science and Technology's DSAI-3201 Machine Learning Course, this project will analyze the UJIIndoorLoc database to explore Wi-Fi fingerprinting techniques for indoor positioning. This first of two parts will overview the concepts behind indoor localization using Wi-Fi Fingerprinting, highlight key feature observation, formulating feature engineering strategies, and validate these strategies with related works.

CCS CONCEPTS • Wi-Fi Fingerprinting • Data Exploration • Machine Learning • Convolutional Neural Networks

**Additional Keywords and Phrases:** Wi-Fi Fingerprinting, Indoor localization

1. Introduction

Indoor localization, also known as indoor positioning system (IPS), is a technology that determines the position of objects or people within enclosed spaces [1]. There are many ways of applying indoor localization; this project will focus on using Wi-Fi Fingerprinting. There are three components to apply indoor localization in this method. (1) Wi-Fi access points that broadcast signals to be received by (2) mobile devices; these signals will be input in (3) an algorithm that processes the data to give an estimated location. Wi-Fi Fingerprinting technique relies on signal strength data which represents the distance of the Wi-Fi access point from a mobile device. It involves training the fingerprinting algorithm to learn the RSSI (Received Signal Strength Indicator) at various points, making markings in the indoor environment creating a fingerprint database [2]. Then, a mobile device can detect signals from the access points and the database matches this current information with the ones stored from the training phase to estimate the device's location. This project aims to create and implement the algorithm using Machine Learning and Deep Learning.

* 1. Dataset

The UJIIndoorLoc dataset is a Multi-Building, Multi-Floor indoor localization database designed to test Indoor Positioning Systems that rely on WLAN/Wi-Fi fingerprinting. It was created by Torres-Sospedra et al [3]. This dataset contains 528 attributes. Of these, 520 are RSSI values, which represent signal intensities from different Wi-Fi Access Points (WAPs). These values help determine the presence and strength of Wi-Fi signals at various locations. The dataset also includes positional attributes: Longitude and Latitude (provide geographic coordinates), Floor and Building ID (indicate where the data was recorded). The other features in this dataset will provide context but they will not be used in the model: Space ID, Relative Position, User ID, Phone ID, and Timestamp.

* 1. Objective

The objective of this project is to develop a machine learning system for indoor localization using Wi-Fi fingerprinting data. The UJIIndoorLoc dataset provides signal strength information from 520 Wi-Fi Access Points (RSSI values), which serve as input features for model training. Two main tasks were carried out in parallel by members of the group, focusing on both classification and regression approaches:

Joy was responsible for the following:

* Classification Model: Developed machine learning classifiers to predict the Building ID and Floor of a given sample using RSSI fingerprints.
* Enhanced Regression Model: Implemented deep learning-based regression models to predict Longitude and Latitude, allowing for fine-grained location estimation within indoor environments.
* In addition to her model development and evaluation part of the report, she was also tasked with polishing up the EDA section and creating standardized functions for preprocessing and evaluation.

Asma was responsible for the following:

* Baseline Regression Model: Built a traditional regression model (e.g., Linear Regression) as a benchmark for predicting geographic coordinates.
* Deep Learning Classification Model: Designed and implemented a Convolutional Neural Network (CNN) to enhance the performance of building and floor classification.
* In addition to her model development and evaluation part of the report, she was also assigned polish the introduction, to do the applied analysis, and write the conclusions

These models contribute to improving indoor positioning systems in scenarios where GPS is unavailable or unreliable, such as shopping malls, office buildings, or underground environments.

1. Data Preprocessing

This section will give a detailed description of the exploratory analysis and rationale behind data processing decisions

* 1. Exploratory data analysis

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: Descriptive analysis of the UJIIndoorLoc dataset

Basic summary statistics were calculated to briefly explore Wi-Fi access points (WAPs). Some WAPs do not detect any signal strength. Some WAPs had no variations in signal strength based on their minimum and maximum values. Some WAPs were noted for removal to limit the dimensionality of the dataset. The number of floor and buildings are noted to be 5 floors and 3 buildings respectively based on the maximum values for these columns. The longitude and latitude has extremely big ranges, suggesting that these are in meters instead of degrees.

A diagram of a number of attributes

AI-generated content may be incorrect.

Figure 2: Correlation matrix of the non-RSSI columns

The correlation matrix shows that the building ID is strongly correlated with longitude and latitude, while expected. This does not provide much information, as the prediction of floor and building will be different from the prediction of longitude and latitude. Meanwhile, floor is not correlated with any other columns, suggesting that Wi-Fi signal strength is essential for predicting floor level as no other attribute can help predict it.

A screenshot of a graph

AI-generated content may be incorrect. A screenshot of a graph

AI-generated content may be incorrect. Figure 3: Pair plots of non-RSSI columns colors by floor and building number

Going deeper into the relationships between columns, the pair plot colored by floor and building ID confirmed that floors are hard to separate based on coordinates, unlike building ID, which shows clear patterns in longitude and latitude. This confirms that floor level cannot easily be inferred from longitude, latitude, and other non-SSI features, hence why Wi-Fi fingerprinting will be used.

A graph of different sizes and colors

AI-generated content may be incorrect.A graph of colored lines

AI-generated content may be incorrect.

Figure 4: Distribution of Records by Selected Features

The distribution plots of non-RSSI columns indicate that Floor 4 has fewer records and Building 2 has more records. This is noted and will be explored in future parts of data exploration. Visualizing the layout of buildings per floor. Coordinates were plotted to visually understand the area of predictions. The scatter plot shows that only building 2 has a 4th floor. Explaining the previous observation that there are less records in floor 4 and more in building 2

A graph with different colored squares

AI-generated content may be incorrect.

A graph of different colored squares

AI-generated content may be incorrect.

Figure 5: Frequency of records on floors and buildings

To investigate further, the frequency distribution of all floors and buildings are plotted. This reveals that even without the 4th floor, Building 2 has the most frequent number of records among all buildings.

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure 6: Distribution plots and RSSI values across all WAP features

To create a better model, it is crucial to understand our RSSI values across all WAPs. To accurately represent that no signal is found on a specific WAP, the no signal value 100 was replaced with -105. The distribution of detected signals is overshadowed by the no signal values. Hence, removing values below -104 reveals the distribution of RSSI values detected. These values are skewed to the right meaning that most detected RSSI value is of low strength.

|  |  |
| --- | --- |
|  | A graph of a bar graph  AI-generated content may be incorrect. |
|  | A graph of different sizes and colors  AI-generated content may be incorrect. |

Figure 7: Plots discovering the propagation of WAP per building and floor

Understanding how signals from WAP behaves, several plots were created to explore how WAPs broadcast through multiple floors and buildings. These plots and analysis showed that WAPs can broadcast frequently to multiple floors, but mostly stay within one building. Most WAPs can be located by their floor and building with their highest RSSI signals, but a few—like WAP008—do not show that clear separation.

Most records in the dataset can receive multiple signals from 10 to 20 WAPs at the same time. This can create a challenge when making predictions as overlapping signals can introduce noise and make it harder to distinguish between locations.

* 1. Data preprocessing decision

The dataset has high dimensionality, including 520 access points—although the 55 WAP columns with zero received signals have been removed—the dimensionality of the dataset remains extremely high which can make model development difficult. And so, experimentation with using principal component analysis was explored. Keeping 95% of the variance removed around 60% of the RSSI columns leaving only 298 of the filtered 465 RSSI columns. Eventually, these PCA columns will not be used in model development as the combined features make the models less interpretable.

Ultimately, the data preprocessing step is reduced to be simple as the dataset is already extremely clean, only a few adjustments must be made for the dataset to be model-ready. The data preprocessing was kept consistent for each model development stage. A standardized function that prepares the data by reading from a specified CSV path, replacing the no-signal found values to be contextualized at -105, removing the never-detected RSSI columns, checking for missing values, and scaling the numerical RSSI columns and coordinate columns for each record.

def load\_and\_clean\_dataset(path):

print(f"Loading {path} and applying the same preprocessing")

df = pd.read\_csv(path)

print(">> The values in WAPxxx columns with 100 is turned into -105")

RSSI\_columns = [col for col in df.columns if col.startswith("WAP")]

df[RSSI\_columns] = df[RSSI\_columns].replace(100, -105)

print(f">> Dropped {len(WAP\_building\_not\_detected)} WAP columns that were never detected.")

df.drop(columns=WAP\_building\_not\_detected, inplace=True)

RSSI\_columns = [col for col in df.columns if col.startswith("WAP")]

print(">> Checking if there are any missing values:", df.isna().sum().any())

print(">> Scaling the numerical columns (RSSI\_columns and Coordinate Columns) with MinMaxScaler")

scaler = MinMaxScaler(feature\_range=(0, 1))

df[RSSI\_columns] = scaler.fit\_transform(df[RSSI\_columns])

scaler\_coords = MinMaxScaler(feature\_range=(0, 1))

df[["LONGITUDE", "LATITUDE"]] = scaler\_coords.fit\_transform(df[["LONGITUDE", "LATITUDE"]])

return df

1. model Development

This section of the report will detail the development of models to predict the location of a record based on RSSI values. Two main groups of predictions will be created: (1) classification models to predict the building number and floor level and (2) regression models to predict the longitude and the latitude of each record.

* 1. Basic Classification Models

The basic model classification started with a training-testing split, 80% for training, 20% for testing. The random state is kept consistent to make the code reproducible. The basic model for classification explored three models, decision tree, support vector machine, and random forest.

Decision tree is one of the simplest models for multiclass classification. It recursively splits the decision-making process into a tree-like model. The predictions are created based on how the record is partitioned. The model itself is implemented using the scikit-learn library. The completed decision tree is as follows.

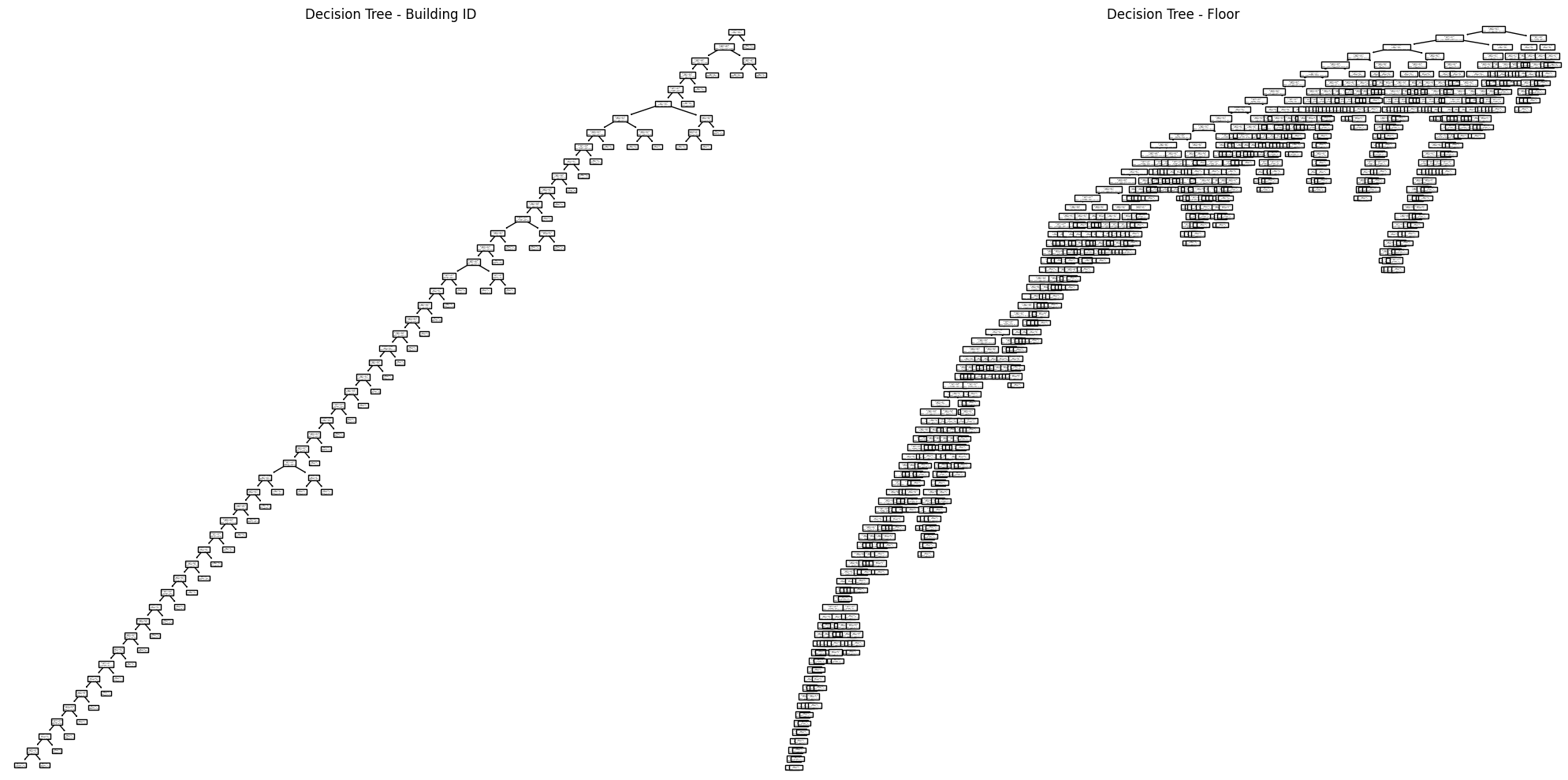


Figure 8: Decision Tree Classifier for Building and Flood

The floor decision tree is much more comprehensive, and fuller compared to the building decision tree, as there are more floors to predict. There are 5 floors in the dataset compared to only 3 buildings. The branches of the trees mostly go to one side, which was a concern at first, but after evaluating the metrics and applying cross-validation with good results, it stopped being a concern. This model is sought after for its simplicity and interpretability. During the initial test, default parameters are preferred, which is the case for this decision tree.

The next basic classification model is a Support Vector Machine (SVM). SVMs can be used for classification and regression tasks. For this basic classification model, a support vector classifier, SVC, will be used. It works by finding an optimal hyperplane that separates the different classes in the space. The features are scaled to help the SVM with its prediction. When training the model initially, a model with PCA columns and a model with the original RSSI columns were both experimented with. However, both produced similar results and keeping the original structure of the features made the model more interpretable. Therefore the model with PCA was not kept. For each model, the default parameters are ideally kept to create a fair comparison between models. For this SVC, the kernel is kept linear because of the high dimensionality of the dataset. The C-control parameter that sets the tradeoff between maximizing margin and minimizing classification error is kept at 1 for a reasonable start for initial testing.

The last basic classification model to predict building and floor locations is Random Forest (RF). Among the three models, RF is the most complicated as it is an ensemble model combining multiple decision trees together to vote on the selected prediction. This Random Forest is also implemented using scikit-learn using default parameters.

The basic models were compared based on their train-test splits, cross-validation, and new dataset performances. Then, one model is chosen for hyperparameter tuning comparing. Hyperparameter tuning was done using differential evolution with a fitness function of F1-macro as opposed to traditional grid search to explore the parameters with direction instead of trying all combinations from a set of values of parameters. Although slower than GridSearch, Differential evolution allows for a more extensive exploration of space. It is optimized by the heuristic nature of the algorithm.

* 1. Advanced Regression Models

Moving on to the Advanced Regression Model. Two deep learning models were compared with each other. A fully connected neural network and a convolutional neural network.

Both deep learning models will involve preparing data loaders using torch sensors and an iterative training on epochs. Several functions are created to avoid repeating code between similar models.

The fully connected Neural Network (NN) was implemented using a class with a default hidden size of 64, a learning rate of 0.001, and a loss function comparing MSE that runs for 100 epochs. A neural network is a machine learning model created to simulate the human brain. It is made up of neurons/perceptron that takes an input and returns a processed result. A fully connected neural network, also known as multilayer perceptron (MLP), is an organized group of perceptron

involving multiple layers. In this type of neural network, every neuron in one layer is connected to every neuron in the next layer. This is good for inputs without spatial or sequential structure.

The Convolutional Neural Network (CNN) is created similarly, integrating grid-like structure to the data to detect patterns. Unlike NN, CNN does not rely on dense connections; instead, it aims to detect patterns in the data, which is useful when there are special patterns in the data. The RSSI features were converted into a 2D vector. CNN was given the same hyperparameters as NN to have a fair comparison between the models.

The deep learning regression models were tuned after evaluating performance through cross-validation and validation on an unseen dataset. The neural network model with the best generalization performance was selected for hyperparameter tuning. As with the earlier classification models, differential evolution was used instead of grid search. While slower, differential evolution is more suitable for deep learning models, where the search space is continuous and high-dimensional. This approach allows for guided exploration across a wider range of values, without being limited to predefined parameter grids. The tuning process focused on optimizing parameters such as hidden layer size, learning rate, and batch size to minimize RMSE.

1. Model Evaluation
   1. Basic Classification

The classification models were evaluated across three main stages: train-test splits, cross-validation, and deployment on a previously unseen dataset (validationData.csv). Metrics focused primarily on F1 macro score, in line with the project’s objective of balanced performance across classes.

* + 1. Initial model metrics

Initial evaluations on an 80/20 split showed consistently high performance across all classifiers.

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure 9: Decision Tree Classifiers Actual vs Predicted

The Decision Tree classifier, despite its simplicity, achieved over 0.99 F1 macro for building classification and approximately 0.975 for floor classification. The model misclassified only 9 out of 3,987 test records for building ID, and floor misclassifications were typically off by one level.

A graph of a number of numbers and a number of squares

AI-generated content may be incorrect.

Figure 10: SVM Classifiers Actual vs Predicted

Support Vector Machine (SVM) classifiers showed marginal improvements over Decision Trees. The building classifier misclassified only 8 records, and the floor classifier achieved a significantly higher F1 macro score of 0.9933.

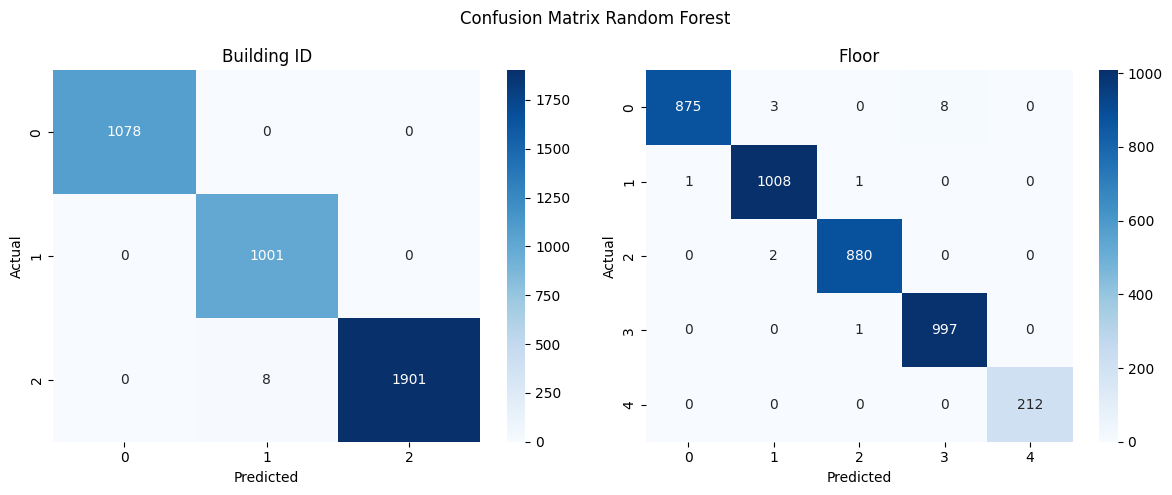
****

Figure 11: Random Forest Classifiers Actual vs Predicted

Random Forest further improved performance, particularly in floor classification, achieving an F1 macro of 0.9965. The ensemble approach provided better generalization with only minor improvements on the already strong SVM results.

* + 1. Cross-Validation Performance

To ensure minimal overfitting of the noise, 10-fold cross-validation was applied to all three classifiers.

Table 1: Average Cross Validation Metrics on Initial Classification Models

| Metric | SVM (Building) | RF (Building) | Tree (Building) | RF (Floor) | SVM (Floor) | Tree (Floor) |
| --- | --- | --- | --- | --- | --- | --- |
| Accuracy | 0.997793 | 0.997743 | 0.996890 | 0.996740 | 0.993178 | 0.974319 |
| Precision Macro | 0.997725 | 0.997962 | 0.996829 | 0.997230 | 0.994047 | 0.974701 |
| Precision Micro | 0.997793 | 0.997743 | 0.996890 | 0.996740 | 0.993178 | 0.974319 |
| Precision Weighted | 0.997810 | 0.997758 | 0.996914 | 0.996764 | 0.993203 | 0.974387 |
| Recall Macro | 0.997902 | 0.997602 | 0.996874 | 0.997208 | 0.993367 | 0.976212 |
| Recall Micro | 0.997793 | 0.997743 | 0.996890 | 0.996740 | 0.993178 | 0.974319 |
| Recall Weighted | 0.997793 | 0.997743 | 0.996890 | 0.996740 | 0.993178 | 0.974319 |
| F1 Macro | 0.997805 | 0.997774 | 0.996839 | 0.997209 | 0.993692 | 0.975401 |
| F1 Micro | 0.997793 | 0.997743 | 0.996890 | 0.996740 | 0.993178 | 0.974319 |
| F1 Weighted | 0.997793 | 0.997742 | 0.996891 | 0.996741 | 0.993177 | 0.974306 |

The results aligned closely with the initial train-test evaluation. Random Forest and SVM consistently produced F1 macro scores above 0.99 for building classification and over 0.99 (Random Forest) and 0.9936 (SVM) for floor classification. Decision Tree floor classification showed slightly more variability across folds, averaging around 0.975.

These results suggested that both SVM and Random Forest were highly stable and reliable across different data splits.

* + 1. Performance on Model Deployment Simulation

Performance on validationData.csv, which had not been seen or used in any model development, revealed slight reductions in performance — particularly for floor classification. While building classifiers maintained performance close to their training evaluations, floor classifiers generally dropped by approximately 10%.

The Decision Tree floor classifier experienced the largest drop, with F1 macro decreasing to 0.7389. This suggests overfitting to the training data and a lack of generalization. In contrast, Random Forest generalized best among the models, achieving an F1 macro of 0.991 for building classification and 0.8845 for floor classification.

* + 1. Hyperparameter Tuning Results

Hyperparameter tuning was applied to Random Forest classifiers using Differential Evolution. This metaheuristic search strategy was used to optimize key parameters with the objective of maximizing F1 macro score.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 12: Basic classification models F1-macro score comparison on test split and validation dataset

However, the improvements were minimal. In the training set, tuned models performed slightly better, but on validationData.csv, performance either remained the same or slightly declined. The tuned building classifier slightly underperformed compared to its default counterpart. The tuned floor classifier achieved a marginal improvement, increasing from 0.8845 to 0.8950 in F1 macro. This suggests that while hyperparameter tuning improved fit to the training data, it may have introduced additional variance or overfitting when evaluated on unseen data.

* 1. Basic Regression

The results from the basic regression are as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |



Figure 13: Predictions from the basic regression models.

* 1. Advanced Classification

The results from the advanced classifications are as follows:

A graph of a number of data

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.A graph of different sizes and colors

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.Figure 14: Predictions from the basic regression models.

* 1. Advanced Regression

The regression models were evaluated on both a train-test split and a completely unseen dataset (validationData.csv). Metrics included RMSE, MAE, and R² score for both longitude and latitude predictions. Cross-validation was also used to assess generalization, and differential evolution was applied to optimize model performance.

* + 1. Initial model performance

Initial testing using the default hyperparameters for Neural Network yielded strong results.

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure 15: Neural Network Regressors Actual vs Predicted

The RMSE for longitude was 0.0315, while latitude had a slightly better RMSE of 0.0233. When scaled back to real-world coordinates, this corresponds to an average prediction error of approximately 12.29 meters in longitude and 6.32 meters in latitude. These error levels appear consistent with what one might expect from indoor positioning systems and are considered low in the context of coordinate-based predictions.

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure 16: Convolutional Neural Network Regressors Actual vs Predicted

The predictions from CNN model were comparable to the previous results. RMSE for longitude improved slightly to 0.0304, whereas latitude RMSE increased slightly to 0.0324. The predictions remained within an acceptable range, showing that both models were effective at predicting spatial coordinates with relatively low error.

* + 1. Cross-Validation Results

Cross-validation confirmed that neither model was overfitting. In fact, one model performed better under cross-validation than on the initial test split. The longitude RMSE improved to 0.0278, and latitude RMSE remained low at 0.0242. The results for the second model remained consistent with the earlier test performance. This reinforced confidence that both models generalized well and were stable across different data splits.

Table 2: Average Cross Validation Metrics on Initial Regression Models

| Metric | NN Longitude | NN Latitude | CNN Longitude | CNN Latitude |
| --- | --- | --- | --- | --- |
| R2 | 0.9921 | 0.9901 | 0.9909 | 0.9842 |
| MSE | 0.00078 | 0.00060 | 0.00091 | 0.00097 |
| RMSE | 0.02785 | 0.02418 | 0.02996 | 0.03084 |
| MAE | 0.01966 | 0.01587 | 0.02074 | 0.02138 |

* + 1. Real-life model deployment simulation

The final validation using validationData.csv tested the generalisation capability of the models on entirely unseen data. The RMSE for the first model was 0.0535 for longitude and 0.0538 for latitude. For the second model, RMSE increased slightly to 0.0565 for longitude and 0.0675 for latitude. These results confirmed that the first model generalised slightly better across datasets, producing more accurate predictions under real deployment conditions.

* + 1. Hyperparameter Tuning Outcomes

Differential evolution was applied to improve prediction accuracy on unseen data. After tuning, the validation RMSE improved substantially. The tuned model achieved an RMSE of 0.0333 for longitude and 0.0433 for latitude. When converted back into real-world distances, this equates to an average error of 13.00 meters and 11.74 meters, respectively. These results demonstrated a measurable improvement from tuning, though the gains were modest in relation to the increased code complexity and experimentation time.

A graph of different sizes and colors

AI-generated content may be incorrect.

Figure 17: Advanced regression models RMSE comparison on test split and validation dataset

USE OF A.I. Tools

While the use of A.I. tools for this project is kept to a minimum, these tools are in the following aspects: creating the pair plots in data exploration, using the stack method in the histograms of RSSI values, combining different plots in subplots with idx, and iterating over rows for detecting RSSI in WAP columns are assisted by OpenAI’s GPT 4o model.

The second part of the project involved the help of ChatGPT 4o in correcting errors in the syntax and working with library functions specifically in writing code for Differential Evolution for both basic and advanced models, debugging and correcting CNN implementation, and creating the objective function for tuning CNN. Applying basic regression and advanced classification and converting from already made code from classifier to regression and vice versa were also aided by ChatGPT 4o.

The formatting of references is also aided by online A.I. tools like Scribbr’s citation generator and, again, GPT4o.

REFERENCES

1. Magda Chelly and Nel Samama. 2009. New techniques for indoor positioning, combining deterministic and estimation methods. In Proceedings of the European Navigation Conference - Global Navigation Satellite Systems (ENC-GNSS 2009), Naples, Italy, 1–12. <https://hal-01367483>.
2. E. Cheng. 2025. Location fingerprinting—What is it, and should you choose it as your IPS technology? Pointr. Retrieved February 5, 2025, from <https://www.pointr.tech/blog/location-fingerprinting-what-is-it-should-you-choose-it>.
3. J. Torres-Sospedra, R. Montoliu, A. Martinez-Us, T. Arnau, and J. Avariento. 2014. UJIIndoorLoc, UCI Machine Learning Repository. <https://doi.org/10.24432/C5MS59>.
4. Android Open Source Project. n.d. Wi-Fi AP/AP concurrency. Android Open Source Project. Retrieved from <https://source.android.com/docs/core/connect/wifi-ap-ap-concurrency>.
5. MetaGeek. n.d. Wi-Fi Signal Strength Basics. MetaGeek. Retrieved from <https://www.metageek.com/training/resources/wifi-signal-strength-basics/>.
6. M. Pierce. n.d. WiFi Signal Strength: A No-Nonsense Guide. TechGrid. Retrieved from <https://techgrid.com/blog/wifi-signal-strength>
7. N. Lang. 2024. What is the MinMax Scaler? Data Basecamp. Retrieved February 16, 2025, from <https://databasecamp.de/en/ml/minmax-scaler-en>.
8. IBM. 2024. What is PCA? Retrieved February 16, 2025, from <https://www.ibm.com/think/topics/principal-component-analysis>.

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)