Report of Advanced Laboratory

Development SW for Indoor Positioning Evaluation

Marcon Lorenzo
DMIF, Computer Science
Università degli Studi di Udine
Udine, Italia
marcon.lorenzo@spes.uniud.it

***Abstract*—This laboratory report presents a software tool to evaluate algorithm and dataset with metrics proposed in the field of Indoor Positioning research area. The user can select appropriate metrics for evaluating indoor positioning algorithms and comparing the performance or evaluate his predictions. The long run goal is to ensure extensibility and customization of algorithms and metrics in the future, with little code update.**

This document is ideally divided into two parts: the first sections present and analyze the problem, reporting the main features and use cases of indoor positioning area, the most cited and known approaches and methodologies proposed to collect observations based on RSS.

The last sections present the design and implementation details of the code and a brief report of the results obtained from the evaluation of predictions.

***Index Terms*—Indoor Positioning, ML, classificazione, KNN, Sklearn, metrics, WKNN**

I. Introduction

In the last years the technologies for collecting data to use for Indoor Positioning has been concentrated on Wifi fingerprints, using RSS (Received Strength Signal) [2], preferring it to other approaches such as ultrasound [11], bluetooth [4] or visible light communication [3]. The already availability and well-known technology of wifi Access Point and its low cost, easy access and performance lead to a standard methodology of collecting, storing and evaluating these fingerprints. In this study, a review of the literature was conducted to identify and analyze the most commonly used metrics in indoor positioning research. These metrics enable the evaluation of the performance of indoor positioning algorithms, essential to assess their accuracy, precision, coverage, and robustness.

The metrics discussed and implemented include accuracy metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Average Positioning Error (APE), which quantify the deviation between the estimated position and the ground truth.

Additionally, precision metrics such as Standard Deviation (SD) of Positioning Error are explored to assess the consistency and reliability of positioning algorithms over time. A short introduction is made for Coverage metrics, including Percentage of Coverage (POC) and Cumulative Distribution Function (CDF), where the dataset collect the logical or physical data of the buildings.

Considering the scope of the laboratory, the observation are treated as *ready to use*, or to better say, *preprocessed* from SNR or outliers features.

Overall, this laboratory report contributes to develop a tool for practitioners and researcher to evaluate the metrics used in indoor positioning research. The insights gained from this analysis can be used in selecting appropriate metrics for evaluating indoor positioning algorithms and comparing their performance.

A. Indoor Positioning data

Indoor prediction, navigation, asset tracking, and context-aware services all depend on Indoor Positioning reasearch area, which is the process of determining an object's location within an indoor environment without the services available outdoor as GPS and GLONASS. Researchers have collected datasets created especially for this purpose to build and evaluate indoor location algorithms and systems. These datasets are useful tools for evaluating and testing various methods and algorithm. A dataset can be collected using different techniques and methodologies, the high availability of hardware and environments lead most studies to use crowdsourced fingerprint logs [5] to enhanced the precision of the true values and the representation of a realistic situation, typically encountered during the roaming of crowd inside an indoor environment. These datasets are carefully curated and structured, but lacking standardization brings complexity and heterogeneity to the problem.

The design and composition of indoor positioning datasets typically involve several key aspects:

1) **Observations:** Mainly RSS signals of the APs in range of a collecting device
2) **Buildings:** Enumeration of the compound or major section of the environment
3) **Floor:** Most of the floors are labeled, others are detected with measuring the meters from the ground (altimeter if available)
4) **Room:** labeled manually
5) **Tile:** logical division of the area, done manually or with computational techniques
6) **Coordinates:** calculating the distance from a zeroing point
7) **Fingerprint:** IDs of an observation

8) **Timestamp:** Time of the observation
9) **Device:** Type or precise description of the device

The coordinates and labeling of building, room, floor and tile serve as ground truth data, the basis for evaluating the performance of positioning algorithms by comparing the estimated positions with the known true positions.

The availability of standard between datasets create and facilitate fair comparisons among different indoor positioning approaches and dataset. Researchers can utilize their own datasets to evaluate their algorithms, validate their findings, and contribute to the collective knowledge in the field.

There are several dataset available for researcher, most of them are public and presented to conferences or competitions, such as IPIN Conference [8].

B. Example of research

The Real-Time Location System (RTLS) [6] is based on Wi-Fi fingerprinting which involves gathering a database of Wi-Fi signal strength values at well-known places and utilizing the values to calculate a specific user's position based on the Wi-Fi signals that their device is receiving. The RTLS system is composed of:

1) Wi-Fi access point network
2) collection of Wi-Fi scanning devices
3) database of Wi-Fi signal strength values and an algorithm for calculating the user's position

The authors first use filtering based on ranking APs on strength signals, then three different selection approaches for the floor estimation, as last the coordinates estimation. The authors use the k-nearest neighbors (kNN) which bases the estimated position on the average position of the neighbors in the database.

The metrics used to evaluate the approaches presented are:

- **Floor Hit Rate:** Percentage of correct floor estimation
- **EVAAL Mean Error:** The cumulative distribution of the errors

The EVAAL Mean Error did not show significant variance between the approaches , meanwhile the first variant of floor estimation reach an hit rate of 94.3% against 91.9% for the other two variants.

The authors then test the system in a real-world indoor environment with various challenges, including obstructions, interference, and dynamic environments. The RTLS system achieved an average positioning error of 2.65 meters, which was among the best results in the competition. They also compare the performance of the k-NN algorithm with other algorithms used in the competition, such as the support vector machine (SVM) and neural network (NN), and show that the k-NN algorithm outperforms these algorithms in terms of accuracy and robustness. The approach presented with a filtering of the building and majority rules for strongest APs selection, implies a reduction of complexity and strongest performance, compared to straight application of KNN and other algorithm to the data.

C. Metrics in literature

Metrics to evaluate prediction for Indoor Positioning Problem can be broadly categorized into four types:

1) **Accuracy**: one of the most commonly used metrics for evaluating indoor positioning systems. It measures the difference between the estimated position and the ground truth position and is usually expressed in terms of the mean or median error. The most commonly used accuracy metrics include Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Average Positioning Error (APE).
2) **Precision**: measures the consistency of the estimated position over time and is usually expressed in terms of the standard deviation of the positioning errors. The most commonly used precision metric is the Standard Deviation (SD) of Positioning Error.
3) **Coverage**: measures the percentage of the indoor environment that can be covered by the positioning system. The most commonly used coverage metrics include Percentage of Coverage (POC), Cumulative Distribution Function (CDF), and Location Stability (LS).
4) **Robustness**: measures the ability of the positioning system to operate in the presence of noise, interference, or other environmental factors that can affect the accuracy and reliability of the estimated position. The most commonly used robustness metrics include Signal-to-Noise Ratio (SNR), Outlier Rate (OR), and Mean Time to Failure (MTTF).

Coverage metrics are used to evaluate the coverage of an indoor positioning system, which refers to the percentage of the indoor area that can be covered by the system. In other words, it measures the ability of the system to detect and locate a user's position in different areas of the indoor environment. The most commonly used coverage metric is the Percentage of Coverage (POC), which is defined as the percentage of the indoor area where the system can provide accurate positioning. This metric is calculated by dividing the area covered by the system by the total area of the indoor environment and multiplying by 100.

The Cumulative Distribution Function (CDF), which measures the percentage of time that the system can locate a user's position accurately. The CDF is calculated by plotting the probability of detection against the detection threshold, where the detection threshold is the minimum signal strength required for the system to detect a signal.

Other coverage metrics include the Average Positioning Accuracy (APA), which measures the average distance between the estimated position and the ground truth position, and the Location Stability (LS), which measures the stability of the estimated position over time.

These metrics are not considered in the scope of the work of this laboratory, where robustness of the system cannot be evaluated and the coverage required more information of the environment.

II. Evaluation Tool

The tool proposed is for standardize the process to evaluate the algorithms for the Indoor Positioning problem. Given a dataset of observations and an algorithm, the tool can deliver to the user a set of evaluation metric, useful to define a hierarchy of performance of the solution proposed in literature.

Preprocessed and standardized datasets are used to reduce possible variations and simplify the applications of the identified algorithms. The application of a standard [1] allows the unification and homogeneity of the implementation, execution and exercise of the tool.

The problem appears as a classification and regression activity, where the classes to which the predictions belong are the physical position indices in space. The first algorithm used and implemented is the well-known *K-Nearest Neighbors* in its weighted and unweighted variants. The difference between *weighted* and *non-weighted* is due to the assignment of a factor to the distance between the observation and K neighbors of the weighted version. The weight helps the algorithm to evaluate more the closest neighbors to the observation, giving less importance to the more distant neighbors.

In addition at the training of model and evaluation of the resulting predictions, it is possible to evaluate directly a user's compliant dataset of predictions with the same metrics developed for the main task.

The features in input to the model are:

- RSS: wireless signal strength recorded by APs within range of the observed device

For the classification activity, the classes to be predicted are:

- Building: building code
- Floor: floor number
- Tile: a logical unit that subdivides a floor

For the regression task, the values to predict are:

- X coordinate
- Y coordinate
- Z coordinate

The training and prediction parameters are searched automatically through the use of specialized libraries such as (Hyperopt, 2022) [7] which searches inside a discrete space of optimal values for minimizing the *loss*, in the case of the KNN algorithm, it searches for the number K of neighbors to consider.

The metric used to measure the distance of nearby nodes can be selected from a drop-down menu:

1) euclidean
2) cosine
3) manhattan
4) grid

Through the tool it is possible to perform three types of tasks:

1) **Training** of an algorithm on standard-compliant dataset
2) **Evaluation** of the model on a dataset of observations
3) **Evaluation Prediction** of a dataset of predictions and respective observations to obtain the prediction metrics

At the end of the operations, the output results are obtained which differ according to the selected task:

- **Training:**
  1) **Plot** of the trend of the parameters (in the case of the example of KNN the number K is obtained as the loss varies),
  2) **Best Models** of regression and classification in .pkl format, to be loaded during the prediction and evaluation phase.
- **Evaluation:**
  1) **Evaluation Dataset** in csv format with features, targets and proposed metrics.
- **Prediction Evaluation:**
  1) **Dataset** in csv format with the evaluation metrics for the predictions uploaded by the user

Evaluation of predictions is performed using metrics selected by the user via interface. In the case of regression, the metrics implemented are RMSE, MAE, MSE, R2, Explained Variance, Median Absolute Error ande Max Error. For the calculation of the RMSE, 2D Mean Error and 3D Error reference was made to the articles that have already analyzed and developed this method, [9], [10].

A. Design

In this section is reported the software design and the project engineering proposal. The objective is to build a local Python script, callable from the working directory, which is composed by:

- source code
- datasets folder
- requirement file

The *main_gui.py* script is callable e it will run the main function of the application, spawning the GUI. The user interface is designed to be used *code free* so it is possible to select parameters and input by buttons and drop down menu.

B. Architecture

The code architecture is divided into two main components:

1) **Offline engine:** Composed of the training part of the algorithms, considered offline, or *cold*, as it searches the parameters of the best ML model and saves it locally to be able to use it in the next evaluation activity , focus of the laboratory.
2) **Online Predictor:** Prediction and evaluation components that calculate the evaluation metrics chosen by the user. For this activity it is possible to choose whether to load an already trained ML model, by loading its parameters in *.pkl* format or by selecting a *.csv* that is *compliant* with the Indoor Database standard containing in addition the fields of predictions to be evaluated.

The chosen algorithm, or model, is then passed through the fitting and prediction cycles, which return the corresponding predictions of the task selected.

C. Use Case

Two main use cases were identified during the design phase:

1) The user wants to train an algorithm already present in the tool, using a Standard Compliant dataset.
2) The user wants to obtain the validation metrics of a prediction set already compliant with framework.

In the first case, the loss function trend, the testing predictions, and the trained model are saved locally and delivered to the user, who can use them for the next prediction and evaluation task.

In the second case, the prediction set must be compliant with the framework, adding the target value for each prediction. The target column must be named as the prediction column with a suffix of the form *_target*. For example, prediction column for regression task are the coordinates coord_x, coord_y, coord_z, so the target column has to be coord_x_target, coord_y_target, coord_z_target.

D. Dataset

Several datasets have been proposed to evaluate indoor positioning algorithms. These datasets contain measurements collected from various sensors, such as Wi-Fi access points, Bluetooth beacons, and magnetic sensors. The datasets are used to evaluate the performance of different indoor positioning algorithms and compare their accuracy, precision, coverage, and latency. In order to evaluate algorithms automatically and independently, it is necessary to apply a standard to the input. Selected datasets must adhere to a framework defined in A Framework for Indoor Positioning including Building Topology, 2022.

For the training activity the input dataset can be selected from a dynamic list or selected by the user:

In the first case the dataset can be added to the folder *datasets*, this is available in the working directory of the script and it must be compliant with the framework proposed.

In the second case, the user can upload a *.csv* file that it has to be structured at least with the following mandatory attributes:

1) fingerprint_id
2) coord_x
3) coord_y
4) coord_z
5) building
6) floor
7) tile
8) multiple features named 'AP-xxx'

The features are the free variable in input to the algorithm. The same observation selected in input are available as output with the prediction. An additional file is created with metrics selected.

III. Models

The model considered is multi-input and multi-output, allowing the possibility of adding input variables to the model and extending the classes to be predicted.

A. K Nearest Neighborhood

The first algorithm made available is a well-known algorithm used both for regression and classification activities. The algorithm uses simple majority voting to label the new data point. KNN allows multiple classification for both *supervised* and *unsupervised* tasks and is based on the similarity principle, which means that similar objects are likely to belong to the same class. In other words, the algorithm works by identifying the K closest to the observation and classifying the point according to the majority class among its K closest neighbors. At start the user can choose the parameter K, which represents the number of nearest neighbors that will be considered to classify a new data point. This value will be tuned using cross-validation or by trying different values of K and choosing the one with the highest accuracy. The most commonly used distance metric is Euclidean distance, but other metrics such as Manhattan or Minkowski distance can also be used to calculate the distance of a new point from the K points. Then, the class of the new data point is selected looking at the majority class among its K nearest neighbors.

There are known limitations, such as being sensitive to the choice of distance metric and the value of K, and being computationally expensive for large datasets.

B. Weighted K Nearest Neighborhood

The second algorithm implemented is the variation of KNN with distance-weighted values, assigning a weight to each neighbor based on its distance.

The weights assigned to each neighbor in WKNN are calculated using a function, such as the inverse of Euclidean distance, which assigns higher weights to closer neighbors and lower weights to farther ones. The weighted sum of the classes or neighbor values is then calculated and the class or value of the new point is determined based on the weighted sum.

Another difference between KNN and WKNN is that WKNN allows for non-uniform weighting of the neighbors, which can improve the accuracy of the algorithm in cases where some neighbors are more important than others. In contrast, KNN assigns equal weights to all neighbors, regardless of their distance to the new data point.

IV. Implementation

A. Modules and Architecture

The code is divided into the following modules:

- main GUI: generates the graphical interface for the user with buttons to select the task. 1
- mode selected: one module for each activity: train, evaluation, evaluation user prediction.
- task GUI: generates the graphical interface for the user with buttons and parameters to load and select data. 3, 6, 5
- predictor: collects the main functions for training and prediction tasks.
- utils data: module with auxiliary functions for data processing, including preprocessing and report creation.

- model class: module for model instantiation and observation fitting and prediction functions.
- metrics: module with function to calculate metric

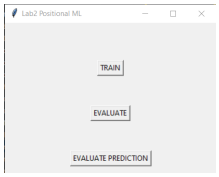

Fig. 1. Main GUI to select the activity

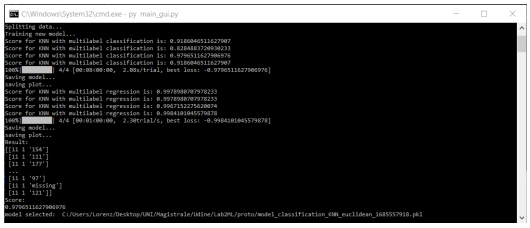

Fig. 2. The prompt for computational details

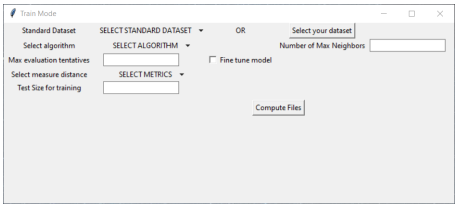

Fig. 3. Training GUI for select training parameters

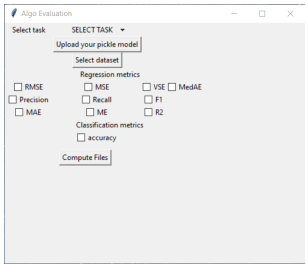

Fig. 4. GUI to evaluate model's prediction

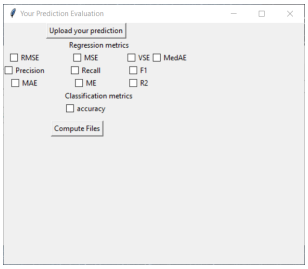

Fig. 5. GUI for user predictions evaluation

The source code is divided in specialize module and the relation between them is design for reuse and easy maintenance. 7

For each activity are available custom interface, where parameters are available to user input or multiple choices.

A hierarchy between modules have been defined:

1) GUI Modules: specialize for GUI and user input
2) Core Modules: divided in *predictor, model class and metrics implementation* implement the functionality of the project
3) Support modules: for supporting activity and relations between modules and libraries. Implements a set of utilities for the project.

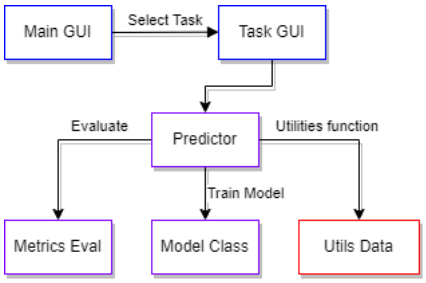

Fig. 6. Module Structure GUI Modules, Core Modules and Support modules

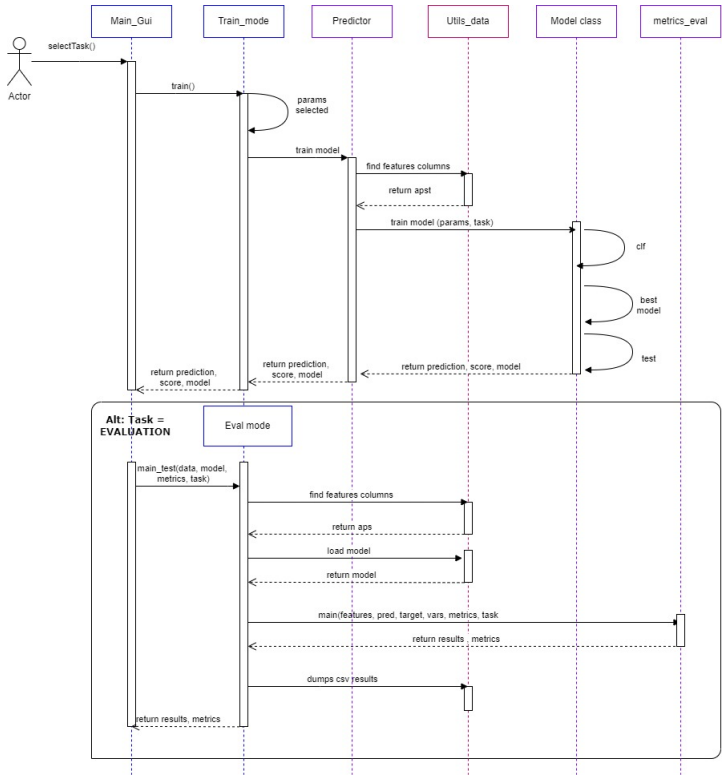

Fig. 7. Flowchart of the computation with GUI Modules, Core Modules and Support modules

V. TRIALS

The algorithms and their corresponding parameters are stored in indexed data structures (dictionaries) that can be extended by adding key-value pairs to the code, implying the recompilation of the application from the source. Depending on the user's choice, the model to be trained is selected and the parameter space is explored and tested with the help of library hyperplot [7]. In the case of selecting the prediction and evaluation task, the model is loaded from the pickle file passed by the user, fit and prediction functions of the sklearn library are called.

In evaluation activity the predictions are uploaded by the user and the metrics selected are handled by the main function of the metric evaluation module. A case selection is performed base on the metrics choosen by the user.

A. Data Manipulation

The dependent variables are bulding, floor, and tile, and are normalized and transformed into ordinal variables (32-bit floating-point numpy) for optimization, an ordinal encoding function is applied after adding a new class *missing* for the missing value of the dataset.

The observation with only *null* values are dropped, because no information is bring by them. The *WAPS* that are *null* are filled with a standard value set to 100. The user can provide a percentage of desired split of the test set, otherwise fixed parameter is then used with a train/test ratio of 80/20.

B. Training

The training phase takes place after a data preprocessing phase, where the classes are encoded into 32-bit floating-point numpy numbers for calculation optimization. The training cycles are managed by the hyperopt library, which searches for the ideal algorithm parameters in the maximum number of for tentative parameters T for loss minimization. Parameters to be used during training are available through GUI, such as maximum K neighbors to use, distance function to use between data points and the percentage of dataset to use as test set. If *finetuning* is selected then it will search from a random $k \in K$ and then iteratively lead to convergence to optimal value of Neighbors. At the end of the phase, the best model is delivered and saved in a .pkl file extension with bar plots of the loss function for each tentative.

C. Metrics

The metrics available to the user are divided in two categories, based on the task selected, *classification* or *regression*. For each predicted point it is calculated the Euclidean Distance from the actual point, then a new attribute *class* is set to 1 if the distance is less then 1.5, otherwise is set to 0. This threshold is defined to transpose the regression problem to a classification problem and calculate the accuracy metrics.

1) Classification Metric: The accuracy, wrong building counter and overall success rate are calculated:

1) **building accuracy**: the ratio between the predicted building and the true values
2) **floor accuracy**: the ratio between the predicted floor and the true values
3) **tile accuracy**: the ratio between the predicted tile and the true values
4) **Succes Rate**: the ratio between the predicted overall labels and the true labels
5) **Wrong Building**: : the difference of total prediction and the correct predictions

2) Regression Metric: For the regression task are available more precision metrics, calculated only if the building is correct, if the building is not predicted correctly, the single value is set to zero.

Listing the already implemented metrics:

1) **RMSE (Root Mean Square Error):** evaluate the accuracy of a prediction. It measures the average magnitude of the differences between the predicted values and the actual values. RMSE calculates the square root of the mean of the squared differences, providing a measure of the overall error between the predicted and actual values. Lower RMSE values indicate better accuracy.
2) **MSE (Mean Square Error):** measures the average squared differences between the predicted and actual values. It is computed by taking the mean of the squared errors. Like RMSE, lower MSE values indicate better accuracy.
3) **VSE (Variance of Squared Errors):** measures the variability of the squared errors between the predicted and actual values. It provides insight into the dispersion of errors and can be useful for assessing the consistency of a prediction model.
4) **MedAE (Median Absolute Error):** calculates the median value of the absolute differences between the predicted and actual values. It is robust to outliers and provides a measure of the typical magnitude of errors. Similar to MAE (Mean Absolute Error), lower MedAE values indicate better accuracy.
5) **Precision:** generally used in classification tasks. It measures the proportion of true positive predictions (correctly predicted positive samples) out of all positive predictions. Precision is useful when the focus is on minimizing false positives. It is considered correct only if the Euclidean distance between the predicted and actual point is less then 1.5. If the threshold is not respected, the prediction is set to 0.
6) **Recall:** also known as sensitivity, is a metric used in binary classification tasks. It measures the ability of a model to correctly identify positive instances out of all the actual positive instances. In simpler terms, it tells us how well the model "recalls" or captures the positive cases.
7) **F1 Score:** harmonic mean of precision and recall and

provides a single value that represents the model's overall performance.

8) **MAE (Mean Absolute Error):** calculates the average absolute differences between the predicted and actual values. It provides a measure of the average magnitude of errors. Lower MAE values indicate better accuracy.
9) **ME (Mean Error):** measures the average difference between the predicted and actual values, regardless of direction. It provides insights into the bias of the prediction model. A ME close to zero indicates that, on average, the model's predictions are unbiased.
10) **R2 (Coefficient of Determination):** statistical metric that represents the proportion of the variance in the dependent variable (actual values) that can be explained by the independent variable (predicted values). It ranges from 0 to 1, with a higher value indicating better performance.
11) **2D Error:** evaluates the accuracy of indoor positioning systems. It measures the Euclidean distance between the predicted position and the ground truth position in two-dimensional space.
12) **Floor Detection Rate:** metric specific to indoor positioning systems that evaluate the system's ability to correctly identify the user's floor level. It measures the proportion of correct floor level predictions out of all the predictions made.
13) **EVAAL Error**: defined in [6],

$$E = Distance(R_i, E_i) + pn_1 \times bFail + pn_2 \times fFail \quad (1)$$

with Distance(Ri,Ei) being the Euclidean distance between the real position and the estimated position, and pn1 and pn2 the penalties associated to incorrect building and floor estimations, respectively. Variable bFail takes the value of 1 for incorrect building estimations and 0 otherwise, while fFail takes the absolute difference between the estimated floor and the correct floor. Values for pn1 and pn2 were set to 50 and 4 meters, respectively.

VI. Results

To evaluate the tool and the training of the algorithm a new dataset, compliant with the format, has been used. The dataset *merged_fingerprints_wifi.csv* is a concatenation of different known Indoor Positioning dataset, like DS1, DS2, IPIN21_track3 and LIB1.

The first run of the tools has been tested with the following parameters:

1) algorithm: KNN
2) max numbers of neighbors to explore: 25
3) max numbers of evaluation cycles: 15
4) dataset: CUSTOM dataset
5) test set at 20%
6) distance measure: euclidean distance

As result, the predictions are reported in *final_results.csv* and the metrics are saved in separated file called *metrics.csv*. The KNN performance are already good after a limited number of trials and the performance of accuracy already reach 85% score with test set, where the best performance are yield by WKNN with over 95% of score with test set.

The reported loss during the iteration of the optimizer are reported as histogram per number of neighbors. The loss values in 8 and 9 are similar and little variance is detected between number of K. The best values for model classifier and regressor are respectively *K=2* and *K=8*.

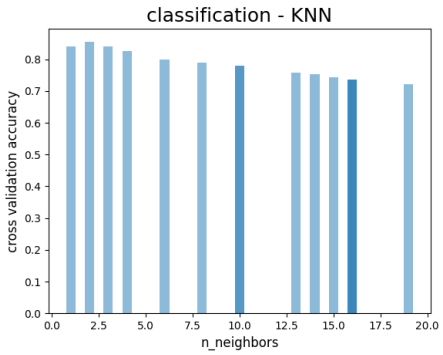

Fig. 8. Loss values per number of neighbors *K*r

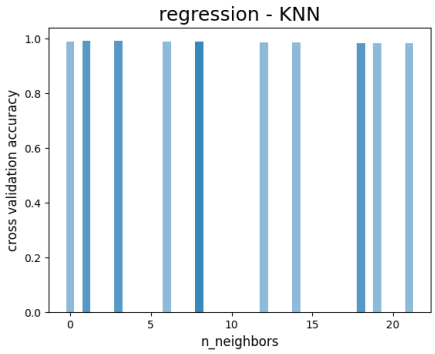

Fig. 9. Loss values per number of neighbors *K*

The values of the metrics are reported in I and the collection of the value for each prediction is available in *final_results.csv* file.

After the evaluation is possible to pass as input a file .csv with the same format of final_results to obtain the metrics directly from the user predictions.

TABLE I
METRICS RESULTS

Test Set	Metric	Value
Custom	RMSE	2.009
Custom	Precision	0.594
Custom	Recall	0.594
Custom	F1	0.594
Custom	MAE	1.485
Custom	MSE	4.037
Custom	R2	-1.206
Custom	EVS	0.0
Custom	MedAE	1.184
Custom	ME	10.535
Custom	2D Error	1.817
Custom	FDP	100.00
Custom	SD	1.352
Custom	EVAAL	17.222

VII. CONCLUSION

The main objective is to offer a ready to go and standard tool to evaluate different dataset and set a baseline for further development on metrics and dataset evaluation for Indoor Positioning. The main characteristics of the tool is the repetition and extensibility, adding dependencies, collection or a single element to the structured data (*dictionary*) of algorithm that can be used and added to the list. An automated and portable solution required some constraints, such as standard input and basic coding skills to update the software.

The repository with source code is available at GitHub Indoor Positioning Tool, this report is also added to the repo.

REFERENCES

[1] Saccomanno Brunello. A framework for indoor positioning including building topology. https://github.com/dslab-uniud/Database-indoor/tree/main/Datasets. Accessed: 2023-05-01.

[2] Genming Ding, Jinbao Zhang, Lingwen Zhang, and Zhenhui Tan. Overview of signals. In *Overview of received signal strength based fingerprinting localization in indoor wireless LAN environments*, pages 160–164, 10 2013.

[3] Trong-Hop Do and Myungsik Yoo. An in-depth survey of visible light communication based positioning systems. *Sensors*, 16(5), 2016.

[4] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *IEEE Journal on Selected Areas in Communications*, 33(11):2418–2428, 2015.

[5] Elena Simona Lohan, Joaquín Torres-Sospedra, Helena Leppäkoski, Philipp Richter, Zhe Peng, and Joaquín Huerta. Wi-fi crowdsourced fingerprinting dataset for indoor positioning. *Data*, 2(4), 2017.

[6] Adriano Moreira, Maria João Nicolau, Filipe Meneses, and António Costa. Wi-fi fingerprinting in the real world - rtls@um at the evaal competition. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2015.

[7] Hyper opt. Hyperopt: Distributed asynchronous hyper-parameter optimization. http://hyperopt.github.io/hyperopt/. Accessed: 2023-05-01.

[8] IPIN org. Ipin conference, international conference on indoor positioning and indoor navigation. https://ipin-conference.org/. Accessed: 2023-05-29.

[9] Nicola Saccomanno, Andrea Brunello, and Angelo Montanari. Let's forget about exact signal strength: Indoor positioning based on access point ranking and recurrent neural networks. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, page 215–224, New York, NY, USA, 2021. Association for Computing Machinery.

[10] Joaquín Torres-Sospedra, Raúl Montoliu, Sergio Trilles, Óscar Belmonte, and Joaquín Huerta. Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems. *Expert Systems with Applications*, 42(23):9263–9278, 2015.

[11] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.