Universitat Oberta de Catalunya

uoc.edu

Development of model based on nearest neighbours with previous knowledge of signal propagation

Dissertation subnitted for the Degree of Master of Data Science by

Vicenç Pio Badia

Supervised by

Joaquín Torres-Sospedra

June, 2022

Abstract

Wi-Fi Fingerprinting is a widely used technique for indoor geolocation. It is based on the measurement of the signals emitted by the near and available Wi-Fi access points. The set of signals received by the device is known as Wi-Fi Fingerprint and is very useful in terms of calculating the position of this object. The algorithm behind this operation is an adaptation of the k-NN, which has been the focus of several studies in the recent years to improve its performance. Because of the own nature of the signal propagation, some interference problems appear due to the obstacles in the environment, both static and dynamic. The present work seeks an implementation of the algorithm with the aim to maximize the accuracy and minimize the computational cost of training the prediction model. To achieve that goal, two main parameters are adjusted in the k-NN algorithm: the k value and the distance metric. The model is trained and evaluated using several heterogenous databases. Finally, a solution called *ensemble* is proposed to improve the results, it consists of combining the outputs of the best metrics to reduce the positioning error.

Keywords

Fingerprint
RSS
Indoor geolocation
Coordinates

List of abbreviations

AP	Access Point
BLE	Bluetooth Low Energy
GPS	Global Positioning System
k-NN	k Nearest Neighbour
RFID	Radio Frequency Identification
RSS	Received Strength Signal
Wi-Fi	Wireless Fidelity

Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Context and work justification

Many applications need the exact position of users for proper working and companies require this technology more and more on its new products and services. The growing world of the Internet of Things is one of the sectors with the most demand. Geolocation in outdoor spaces is usually easy to resolve by using GPS systems embedded into all smartphones. However, geolocation in indoor environments comes with more limitations because GPS signals are not strong enough inside a building or in narrow spaces such a street with tall buildings surrounding. Another inconvenience is that GPS cannot distinguish at which floor is placed the device.

There are currently several indoor positioning techniques based on different technologies: Wi-Fi, BLE, RFID, etc. In our case of study, we focus on Wi-Fi fingerprinting with k-NN algorithm to create a prediction model using the measurements of the signals emitted by the near and available Wi-Fi access points. Related studies approaching the problem show results with some accuracy errors between the predicted location and the actual location of the device measured. The aim of this work is to implement a k-NN to improve the accuracy and reduce the computational cost when training the model. The focus is on the treatment of the signal propagation conditions and the handling of the errors caused by the present obstacles in the environment.

1.2 Main objectives

The main objective of the project is to design and implement a predictive model based on the k-NN algorithm with previous knowledge of the signal propagation. The partial objectives are the following:

- Data cleaning, data filtering and identification of outliers.
- Implement and train the k-NN model

- Compare results with different k values and distance metricd to optimize the results in this context.
- Find the configuration with best accuracy and lowest computational cost.

1.3 Personal motivation

My interest in engineering and telecommunications comes from many years ago when studying at high school. The field of information technologies was my choice when starting the bachelor at university and for the jobs I've worked during the last years. The Master of Data Science opened a new window of knowledge for me, and I believe this topic is a good field of study to do a deeper insight. I have already worked in projects involving indoor positioning but using other technologies such as UWB, RFID and RF but never Wi-Fi. Because all this reasons I think this project is a good opportunity for me to finalize this master's degree.

1.4 Approach and methodology

The source of the data that will be used comes from a database filled with Wi-Fi signals collected from indoor spaces. That data includes the RSSI of each of the signals as well as other information. The first step will be to review the state of the art about Wi-Fi Fingerprinting and related recent studies. The next step is the design and implementation of the algorithm: the first stage will be the data preparation including data cleaning and selection of the interesting data from the training data base. Then the algorithm will be tested using different distance metrics to find the solution that fits best. The same process will be done with the data representation. The test data comes from a different source of the training data to simulate the real conditions and evaluate the algorithm. The development of the project will be coded in Python on the Jupyter Notebook environment.

1.5 Calendar planification

The project is divided in seven stages to separate the main tasks:

- Definition and planification. From 16/02 to 27/02.
- State of the art revision. From 28/02 to 13/03.
- Design and implementation. From 14/03 to 15/05.
- Writing of the thesis (first delivery). From 16/05 to 29/05.

- Writing of the thesis (final delivery). From 30/05 to 05/06.
- Preparation of defense. From 06/06 to 12/06.
- Public defense. From 13/06 to 24/06.

1.6 Summary of obtained products

The obtained product is basically a library of functions with the following implementations:

- Calculation of k-NN and position estimation.
- Positioning error of the prediction.
- Average error of positioning errors.
- Distance metrics comparison.
- Ensemble of selected distance metrics.
- ECDF and scatter plots of the positioning errors.

1.7 Summary of the chapters

This project includes five main chapters that are described as follows:

- Related work: first explains the subareas of the project development, then the possible applications and finally presents a list of similar investigations related to the topic of this project.
- Materials and methods: this chapter introduces the data structure used along the project with examples. It also describes the main functions that compose the k-NN algorithm.
- Description of the proposed approach: it is explained the process and methodology carried out during the project. It details the incremental implementation of the algorithm through different stages and the difficulties encountered.
- Experimental results and discussion: this chapter presents the outcomes obtained from the model and does some comparison and interpretation using plots and tables.
- Conclusions and future works: it explains what initial objectives are achieved or not and why. It also revises the methodology and planification followed and finally states some future lines of investigation related to this topic.

Chapter 2

Related work

In this section, we first detail the different subareas contained within the scope of this work and the summary of the main tasks for each of them. Then we review several areas where this project could be applied and give value to possible applications. Finally, we present the state of the art of similar investigations.

2.1 Subareas within the scope of my work

The scope of the project includes several subareas that will be approached during the implementation of the algorithm. They can be divided as follows:

- Data cleaning: is the first step when working with large datasets. Includes the selection of the observations of interest and the possible transformations of some data if needed. A good process of data cleaning is fundamental for the performance of the algorithm because it helps to reduce the computational cost of the training.
- Algorithm implementation: consists of selecting the most relevant fingerprints focusing on the signal propagation. It is the main work of the project, and the goal is to implement an “improved” k-NN that predicts the position of the devices with the best accuracy.
- Train and test: a training database will be used to train the model with the best parameters to obtain the best results. The testing of the algorithm will be done with a separate database filled with testing data.

2.2 Possible applications in one area

In the recent years, indoor positioning has grown as one of the most demanded technologies by companies working in sectors such as IoT and Industry 4.0. We can find multiple examples of applications for indoor location, from the navigation of hospitals, airports, parking garages and shopping malls, for example, to navigational

aids for the blind and visually impaired, targeted advertising, mining, and disaster response.

The use of Wi-Fi for the positioning comes with two main advantages: all these applications can be embedded in an average smartphone since all of them include Wi-Fi as standard. The other advantage is that most of the buildings have their own Wi-Fi access points so there is no need to install extra hardware to use these applications.

2.3 Similar investigations

The last part of this section is to present some of the most relevant works of the state of the art in relation with our field of study. All these articles will be useful to compare results with our own work at the end of the project.

Zhenghua Chen et al., (2019) present a local feature-based deep long short-term memory (LF-DLSTM) approach. The local feature extractor attempts to reduce the noise effect and extract robust local features. The DLSTM network is able to encode temporal dependencies and learn high-level representations for the extracted sequential local features.

Xudong Song et al., (2019) implement a novel classification model and a novel positioning model by combining a Stacked Auto-Encoder (SAE) with a one-dimensional Convolutional Neural Network (CNN). The SAE is utilized to precisely extract key features from sparse Received Signal Strength (RSS) data while the CNN is trained to effectively achieve high accuracy in the positioning phase.

Jianwei Niu et al., (2015) implement weighted KNN algorithm to assign different weights to APs and achieve room-level localization. To obtain the absolute coordinate of users, they design a novel MDS algorithm called MDS-C (Multi-Dimensional Scaling with Calibrations) to calculate coordinates of interested locations in the corridor and rooms, where anchor points are used to calibrate absolute coordinates of users.

Hurkan M. Aydin et al., (2021) propose to use feature selection methods along with the K-nearest neighbours (KNN) classification and regression algorithms in order to create a simple and swift location positioning system. The evaluation of various feature selection methods shows that computation times for positioning can be reduced by 75% using feature selection.

Jing, Hao et al., (2014) design a collaborative Wi-Fi fingerprint training (cWiDB) method that enables the system to perform inertial measurement based collaborative positioning or Wi-Fi fingerprinting alternatively according to the current situation. It also reduces the time required for training the fingerprint database. Different database training methods and different training data size are compared to demonstrate the time and data required for generating a reasonable database. Finally, the fingerprint positioning result is compared which indicates that the cWiDB is able to achieve the same positioning accuracy as conventional training methods but with less training time and a data adjustment option enabled.

Mok, Esmond and Cheung, Bernard K. S. (2013) propose a Wi-Fi positioning algorithm based on neural network modelling of Wi-Fi signal patterns. This algorithm is based on the correlation between the initial parameter setting for neural network training and output of the mean square error to obtain better modelling of the nonlinear highly complex Wi-Fi signal power propagation surface. The test results show that this neural network-based data processing algorithm can significantly improve the neural network training surface to achieve the highest possible accuracy of the Wi-Fi fingerprinting positioning method.

Finally, Rui Zhou et al., (2016) present a Wi-Fi fingerprinting algorithm based on Support Vector Machines (SVM), which combines SVM classification and regression to model the unknown relationship. During sampling and training, the indoor area is partitioned to subregions and the nonlinear relationship between signal fingerprints and locations as well as subregions are established. For positioning, SVM classifiers first determine the subregion that the mobile device is in, then SVM regression estimates the exact coordinate on the basis of classification result.

Chapter 3

Materials and methods

This chapter describes the structure of the data and the environment used along the project with details and examples. It also gives introduction to the main methods implemented to compose the k-NN algorithm.

3.1 Test environment

The data used in the experimentation [X] comes from different areas of University Jaume I (València). It corresponds to several rooms of several floors and buildings and the area includes obstacles that can block or disturb the RSS signals.

3.2 Datasets structure

The data collected is stored in files with CSV format. These are the datasets used in this project:

Filename	Train	Test	Total
DSI1	1369	348	1717
DSI2	576	348	924
LIB1	576	3120	3696
LIB2	576	3120	3696
MAN1	14300	460	14760
MAN2	1300	460	1760
SIM0001	10710	1000	11710

Table 3.1: List of datasets used for training and testing the model

Each dataset is divided in four separate CSV files following this structure:

- Training coordinates
- Training RSS
- Test coordinates
- Test RSS

On one hand, the coordinates files include: x, y, z, floor, building. For this work, and with the purpose to simplify the computation, the variables floor and building are avoided.

On the other hand, the RSS files include the actual fingerprint, which is a vector of RSS. The vectors of RSS are represented in dB and the value 100 indicates signal not received. The length of the fingerprints varies from one dataset to another. An example of an RSS row could be:

100,-79,100,100,100,100,100,100,100,100,100,100,100,100,-84,100,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,-72,-72,100,-84,100,100,100,100,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,100,100,100,100,100,100,-85,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100

Notice that most of the values are 100, it indicates that the signal of this particular Wi-Fi AP is not received by the device measuring.

An example of a coordinates row could be:

73.75653190807489,54.63366094694744,3.7,1,1

That can be translated as:

Variable	**Value**
x	73.75653190807489
y	54.63366094694744
z	3.7
floor	1
building	1

Table 3.2: Coordinates file row example

3.3 Methods

In this section are explained the methods implemented during the project:

K-NN

It is a supervised algorithm used for both classification and regression. In our case is used for regression and it consists basically of finding the average value of the k nearest neighbours for each test fingerprint to obtain the estimated position coordinates.

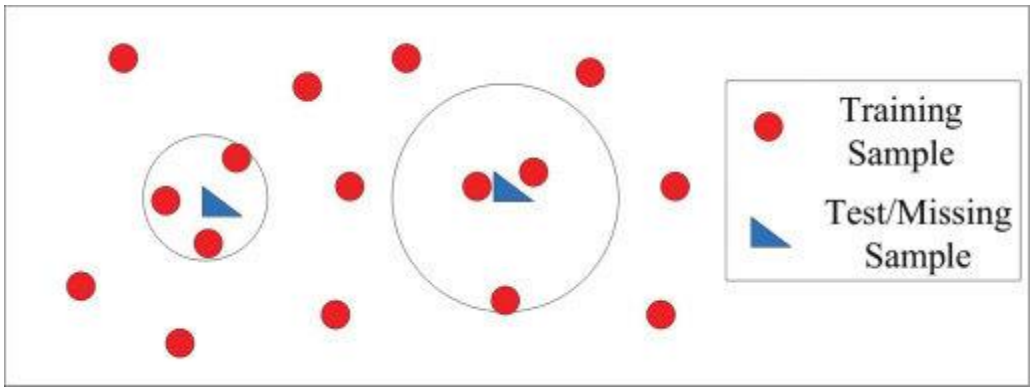

Figure 3.1: Scenario with 2 test samples and k=3

The procedure to find the nearest neighbours follows these steps:

- The circles cover 3 training samples each, it indicates that in this example k=3.
- The first step is to calculate the distance between one test fingerprint (blue triangle) and all the training samples of the dataset (red circles). The methods to calculate the distance will be explained on the following sections.
- Once obtained the list of distances, the list is sorted and only the first k samples are selected. Those k samples are the k nearest neighbours.
- From these k samples (fingerprints), the next step is to calculate the mean value of their corresponding coordinates.
- The result from the previous calculation is the estimated position of the initial test sample.
- This procedure is repeated for each test fingerprint. The final output is a list of estimated positions corresponding to each of the fingerprints.

Positioning error

The positioning error is the shortest distance (Euclidean) between the estimated position and the real position of a sample. The formula of the Euclidean distance in two dimensions is:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

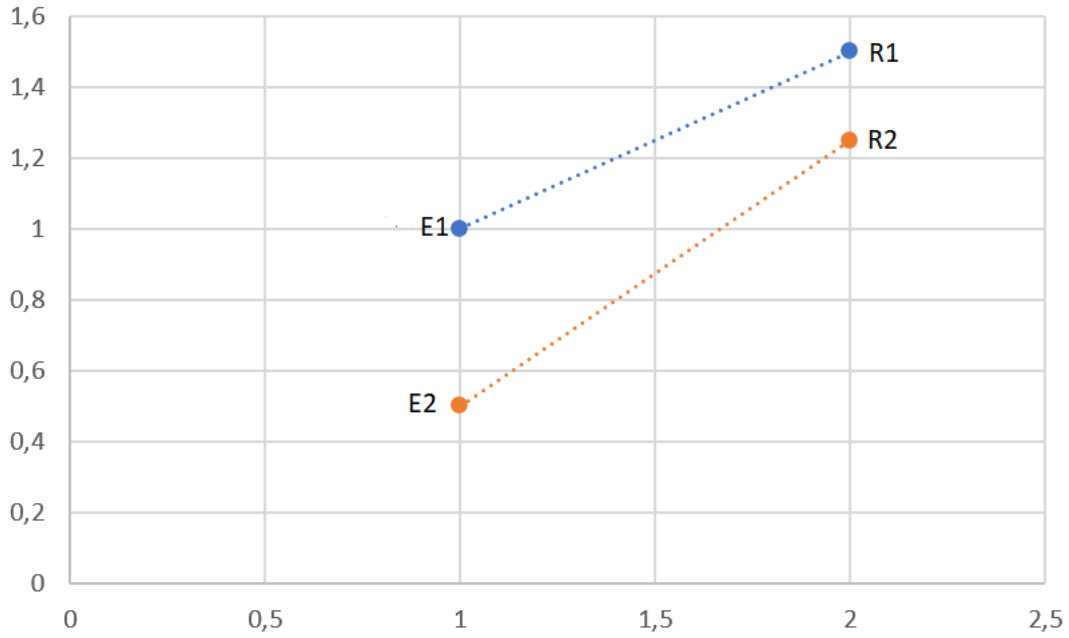

Figure 3.2: Minumum distance between coordinates

In the previous figure is shown an example of two estimated coordinates by the algorithm (E_1 and E_2) joined with a line to the real position of these coordinates (R_1 and R_2). The line indicates the Euclidean distance between them. In this case, the coordinates value is:

$$E_1 = (1, 1)$$
$$E_2 = (1, 0.5)$$
$$R_1 = (2, 1.5)$$
$$R_2 = (2, 1.25)$$

Now the distance error of the two pairs of coordinates can be calculated:

$$d_1(E_1, R_1) = \sqrt{(2-1)^2 + (1.5-1)^2} = 1.12$$

$$d_2(E_2, R_2) = \sqrt{(2-1)^2 + (1.25-0.5)^2} = 1.25$$

The previous process is done for each one of the predicted positions by the k-NN. During the development of the project, the evaluation of the algorithm has been based on the average error of the predicted positions. The average error is calculated using the mean value of the positioning errors for all the positions estimated.

Then the average error of the example is:

$$AvEr = \frac{1.12 + 1.25}{2} = 1.18$$

Chapter 4

Description of the proposed approach

The objective of the project is to find the best configuration of the k-NN algorithm to obtain a good accuracy and average error. To achieve that goal, the followed methodology consists of an incremental implementation of the algorithm. That means starting with a basic k-NN on the first tests and then adding several features to end up with the best results. In this chapter is explained this process and the detail of every step focusing on the technical perspective.

4.1 Basic k-NN using Euclidean distance

The basic k-NN implementation includes the next modules:

- k-NN: this is the core of the algorithm. As previously explained, the main functionality of the algorithm is to iterate the fingerprints of the test RSS data to find the k nearest neighbours within the train RSS data for each one of them. This first implementation serves to obtain a list of fingerprints indexes, the next step is to link them to its corresponding coordinate.

- Predict positions: using the fingerprint indexes, the algorithm finds the corresponding coordinates in the train coordinate dataset and calculates the mean value to end up with the estimated position for each fingerprint.

- Calculate error: the list of predicted positions needs to be contrasted with the actual position stored in the test coordinates dataset. The two lists of positions are compared, and the distance error is calculated using the Euclidean distance. That distance errors can be used to evaluate the model using ECDF and scatter plots, this part is explained in the next chapter.

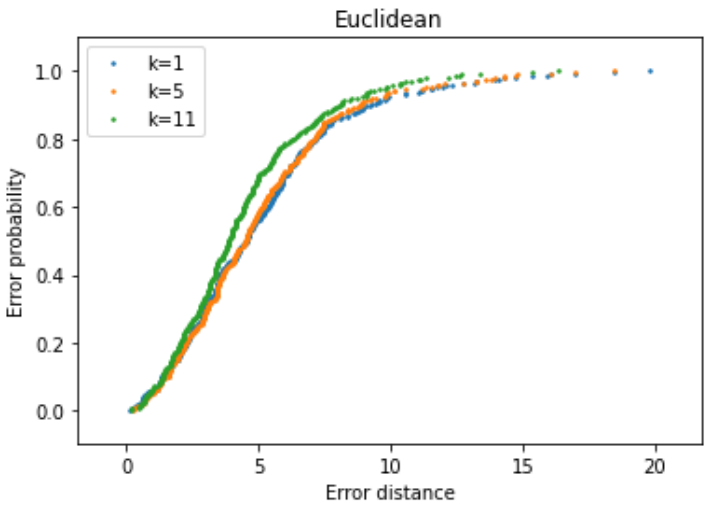

Figure 4.1: Example of the positioning error seen in a ECDF plot

At this point, the basic implementation of the algorithm is done. The next steps describe the process followed during the project to optimize the accuracy and reduce the positioning error.

4.2 Distance/similarity metrics

The basic implementation uses the Euclidean distance to find the nearest neighbours. However, it is a good practice to compare the average errors obtained by the execution of the algorithm using different distance metrics. This is the list of the metrics studied in this project grouped by their respective families:

Lp Minkowski family	
Euclidean L2	$d_{Euc} = \sqrt{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^2}$
Minkowski Lp	$d_{Mk} = \sqrt[p]{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^p}$
City block	$d_{CB} = \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$
Chebyshev	$d_{Cheb} = \max_i \lvert P_i - Q_i \rvert$

L1 family

Sørensen	$$d_{sor} = \frac{\sum_{i=1}^{d} \mid P_i - Q_i \mid}{\sum_{i=1}^{d} (P_i + Q_i)}$$
Gower	$$d_{gow} = \frac{1}{d} \sum_{i=1}^{d} \frac{\mid P_i - Q_i \mid}{R_i} = \frac{1}{d} \sum_{i=1}^{d} \mid P_i - Q_i \mid$$
Soergel	$$d_{sg} = \frac{\sum_{i=1}^{d} \mid P_i - Q_i \mid}{\sum_{i=1}^{d} \max(P_i, Q_i)}$$
Kulczynski d	$$d_{kul} = \frac{\sum_{i=1}^{d} \mid P_i - Q_i \mid}{\sum_{i=1}^{d} \min(P_i, Q_i)}$$
Lorentzian	$$d_{Lor} = \sum_{i=1}^{d} \ln(1 + \mid P_i - Q_i \mid)$$
Canberra	$$d_{Can} = \sum_{i=1}^{d} \frac{\mid P_i - Q_i \mid}{P_i + Q_i}$$

Intersection family	
Intersection	$$d_{non-IS} = 1 - s_{IS} = \frac{1}{2} \sum_{i=1}^{d} \mid P_i - Q_i \mid$$
Wave Hedges	$$d_{WH} = \sum_{i=1}^{d} \left(1 - \frac{\min(P_i, Q_i)}{\max(P_i, Q_i)}\right) = \sum_{i=1}^{d} \frac{\mid P_i - Q_i \mid}{\max(P_i, Q_i)}$$
Czekanowski s	$$s_{Cze} = \frac{2 \sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$$
Czekanowski d	$$d_{Cze} = 1 - s_{Cze} = \frac{\sum_{i=1}^{d} \mid P_i - Q_i \mid}{\sum_{i=1}^{d} (P_i + Q_i)}$$

Motyka s	$$s_{Mot} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$$
Motyka d	$$d_{Mot} = 1 - s_{Mot} = \frac{\sum_{i=1}^{d} \max(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$$
Kulczynski s	$$s_{Kul} = \frac{1}{d_{Kul}} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} \mid P_i - Q_i \mid}$$
Ruzicka	$$s_{Ruz} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} \max(P_i, Q_i)}$$
Tanimoto	$$d_{Tani} = \frac{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - 2\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - \sum_{i=1}^{d} \min(P_i, Q_i)}$$ $$= \frac{\sum_{i=1}^{d} (\max(P_i, Q_i) - \min(P_i, Q_i))}{\sum_{i=1}^{d} \max(P_i, Q_i)}$$

Inner product family	
Inner Product	$$s_{IP} = P \bullet Q = \sum_{j=1}^{d} P_i Q_i$$
Harmonic mean	$$s_{HM} = 2\sum_{i=1}^{d} \frac{P_i Q_i}{P_i + Q_i}$$
Cosine	$$s_{Cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2} \sqrt{\sum_{i=1}^{d} Q_i^2}}$$

Kumar-Hassebrook	$s_{Jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$
Jaccard s	$s_{Jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$
Jaccard d	$d_{Jac} = 1 - s_{Jac} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$
Dice s	$s_{Dice} = \frac{2\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$
Dice d	$d_{Dice} = 1 - s_{Dice} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$

Fidelity family or Squared-chord family	
Fidelity	$s_{Fid} = \sum_{i=1}^{d} \sqrt{P_i Q_i}$
Bhattacharrya	$d_B = -\ln \sum_{i=1}^{d} \sqrt{P_i Q_i}$
Hellinger	$d_H = \sqrt{2\sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2}$ $= 2\sqrt{1 - \sum_{i=1}^{d} \sqrt{P_i Q_i}}$
Matusita	$d_M = \sqrt{\sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2}$ $= \sqrt{2 - 2\sum_{i=1}^{d} \sqrt{P_i Q_i}}$
Squared-chord	$d_{sqc} = \sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2$

Squared L2 family	
Squared Euclidean	$d_{sqe} = \sum_{i=1}^{d} (P_i - Q_i)^2$
Pearson χ^2	$d_P(P,Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i}$
Neyman χ^2	$d_N(P,Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}$
Squared χ^2	$d_{SqChi} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$
Probabilistic Symmmetric χ^2	$d_{PChii} = 2\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$
Divergence	$d_{Div} = 2\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{(P_i + Q_i)^2}$
Clark	$d_{Clk} = \sqrt{\sum_{i=1}^{d} \left(\frlac{\lvert P_i - Q_i \rvert}{P_i + Q_i} \right)^2}$
Additive symmetric χ^2	$d_{AdChi} = \sum_{i=1}^{b} \frac{(P_i - Q_i)^2 (P_i + Q_i)}{P_i Q_i}$

Shannon's entropy family	
Kullbac Leiber	$d_{KL} = \sum_{i=1}^{d} P_i \ln \frac{P_i}{Q_i}$
Jeffreys	$d_J = \sum_{i=1}^{d} (P_i - Q_i) \ln \frac{P_i}{Q_i}$
K Divergence	$d_{Kdiv} = \sum_{i=1}^{d} P_i \ln \frac{2P_i}{P_i + Q_i}$
Topsoe	$d_{Top} = \sum_{i=1}^{d} \left(P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right)$
Jensen-Shannon	$d_{JS} = \frac{1}{2} \left[\sum_{i=1}^{d} P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + \sum_{i=1}^{d} Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right]$
Jensen difference	$d_{JD} = \sum_{i=1}^{b} \left[\frac{P_i \ln P_i + Q_i \ln Q_i}{2} - \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2} \right) \right]$

Combinations

Taneja	$d_{TJ} = \sum_{i=1}^{d} \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2\sqrt{P_i Q_i}} \right)$
Kumar-Johnson	$d_{KJ} = \sum_{i=1}^{d} \left(\frac{(P_i^2 - Q_i^2)^2}{2(P_i Q_i)^{3/2}} \right)$
AvgL	$d_{ACC} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert + \max_i \lvert P_i - Q_i \rvert}{2}$

Vicissitude	
Vicis-Wave Hedges	$d_{emanon1} = \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{\min(P_i, Q_i)}$
Vicis Symmetric 1	$d_{emanon2} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\min(P_i, Q_i)^2}$
Vicis Symmetric 2	$d_{emanon3} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\min(P_i, Q_i)}$
Vicis Symmetric 3	$d_{emanon4} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\max(P_i, Q_i)}$
Min-Symmetric	$d_{e6} = \min \left(\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}, \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i} \right)$
Max-Symmetric	$d_{e5} = \max \left(\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}, \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i} \right)$

Table 4.1: Distance/similarity metrics equations

As a result of the execution of these set of metrics, some errors appeared due to the format of the input data. These errors were always provoked by divisions by 0 and square root and logarithm of negative numbers. The specific implementation of some of the metrics was incompatible with the format of the fingerprints.

Divisions by 0

To fix the issue of divisions by 0, the solution is to add a numeric factor to all the divisors to prevent them to be 0. The factor is set to a very small value (10^{-7}) so that it doesn't affect the result of the operation. In the following example we can see the Neyman distance with the factor added to the divisor:

```
FACTOR = 1e-7

def neyman(P, Q):
    P=np.array(P,dtype=float); Q=np.array(Q,dtype=float)
    return sum((P-Q)**2/(P+FACTOR))
```

Figure 4.2: Neyman distance with the added numeric factor

Data representation

To deal with the other errors occurred, three alternative solutions are proposed for the data representation: min-max normalization, absolute value and increment value:

- Min-max normalization: the concept of min-max normalization stands for finding the lowest and greatest value of the entire dataset and set them to 0 and 1 respectively. From that point, the rest of the values are scaled so that the whole dataset ranges from 0 to 1.

  An important detail here is that the scaler must use the RSS values of train and test altogether to find the general minimum and maximum. Otherwise, the scale would be wrong, and that is something that happened to the present implementation and produced some incorrect outputs before it was correctly solved.

- Absolute error: this one is the simplest alternative and allows to convert all the values to positive numbers.

- Increment: this solution involved making the decision of what value choose for the increment. The datasets used has slightly different range of values but, in general, the approximate minimum value is about -95. Based on that, the increment value was set to 100 to ensure all data to be above 0.

These conversions are all three made just before calculating the distance between the two fingerprints. The function that makes this operation selects the conversion wanted on every execution depending on the parameters set each time.

It is important to mention that the positioning error obtained from the k-NN suffers an alteration after executing the algorithm with one of the mentioned options. In some cases, the positioning error increases and in other cases decreases, it depends on several factors: the characteristics of the dataset, the k value, etc.

The only conversion that should not alter the results is the min-max normalization because it maintains the scale of the data so the distance computation should not be modified. However, in this case it does not happen. That is because the train and test datasets are separated, and the scaler must be set to either one of them and later apply the transformation to both of them. If, for instance, the scaler is set to the train data, it could happen that in the test dataset exist some lower value that later will be converted to negative when transformed with the training scale. A possible solution could be to join both datasets before the min-max scaling but in terms of facilitate the coding, the solution implemented is to set to 0 the negative values and set to 1 the values greater than 1 after the normalization is done. That is reason why the average errors obtained with and without normalization are slightly different.

Here we can see a results comparison between the three data representations:

Distance metric	**Normalization**	**Absolute value**	**Increment**
Euclidean	6,52	5,04	6,46
MinkowskiL3	6,59	4,95	6,56
MinkowskiL4	6,49	5,04	6,47
MinkowskiL5	6,48	5,58	6,53
City Block	6,34	6,25	6,38
Chebyshev	21,76	6,62	21,18
Sorensen	25,76	4,95	26,65
Gower	6,34	7,67	6,38
Soergel	6,28	26,65	6,28
Kulczynski d	6,28	4,95	6,28
Lorentzian	6,18	4,99	5,91
Canberra	5,82	4,99	5,81
Intersection	6,34	5,66	6,38
Wave Hedges	5,83	4,8	5,77
Czekanowski s	6,28	4,95	6,28
Czekanowski d	46,62	4,7	47,32
Motyka s	6,28	4,99	6,28
Motyka d	46,62	51,34	47,32
Kulczynski s	6,28	4,99	6,28
Ruzicka	6,28	51,34	6,28
Tanimoto	6,28	4,99	6,28
Inner Product	39,32	4,99	39,76

Harmonic	28,16	4,99	28,73
Cosine	6,43	40,68	6,42
Kumar-Hassebrook	6,73	57,31	6,51
Jaccard s	6,73	5,06	6,51
Jaccard d	39,25	5,1	47,16
Dice s	6,47	5,1	6,51
Dice d	45,94	46,01	47,16
Fidelity	40,14	5,1	40,56
Bhattacharrya	40,53	46,01	41,54
Hellinger	6,44	40,68	6,29
Matusita	6,44	57,31	6,29
Squared-chord	6,44	5,01	6,29
Squared Euclidean	6,52	5,01	6,46
Pearson	16,38	5,01	13,91
Neyman	13,51	5,04	8,75
Squared	6,47	5,11	6,45
Probabilistic Symmmetric	6,47	4,87	6,45
Divergence	6,36	5,01	6,09
Clark	6,36	5,01	6,09
Additive symmetric	11,81	4,77	7,34
Kullbac Leiber	24,71	4,77	24,84
Jeffreys	6,94	4,99	6,52
K Divergence	39,94	57,31	42,63
Topsoe	6,46	5	6,39
Jensen-Shannon	6,46	57,31	6,39
Jensen difference	6,46	5,01	6,39
Taneja	7,33	5,01	6,61
Kumar Johnson	15,08	5,01	8,35
AvgL	6,35	5	6,41
Vicis-Wave Hedges	12,87	4,98	7,18
Vicis Symmetric 1	15,49	4,97	10,25
Vicis Symmetric 2	12,98	4,76	7,24
Vicis Symmetric 3	6,27	4,9	6,43
Min-Symmetric	15,07	4,91	12,52
Max-Symmetric	12,81	5,02	8,46
Mean value	**13,96**	**13,46**	**13,45**

Table 4.2: Results from DSI dataset for k=1

Ensemble

The *ensemble* is the final improve made to achieve the objective of optimize the k-NN results. It consists of selecting the best performing metrics and calculate the centroids of the predicted positions. The strength of this procedure is that it combines several results and the possible high errors of one metric prediction are compensated by the

others. It could happen that some particular predictions of the ensemble were worse than the original prediction from the metric, but the average error will always be better. The output is the final predicted position.

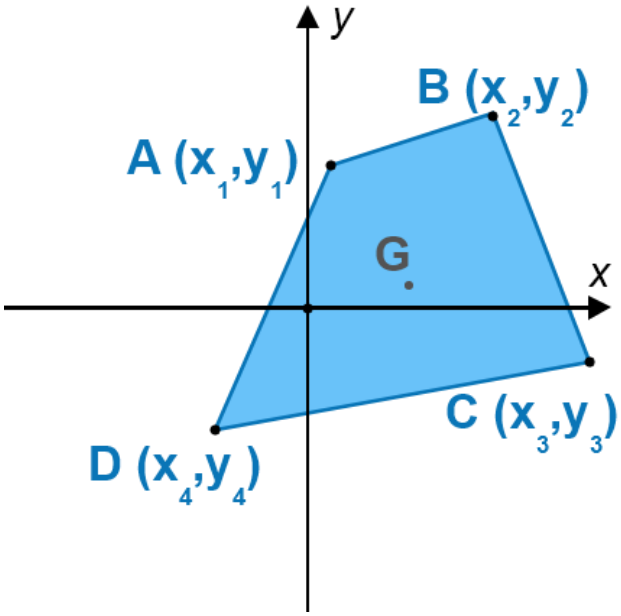

Figure 4.3: Centroid (G) of four coordinates in a 2-axis plot

The figure 4.2 shows the centroid of four coordinates in a 2-dimensional space. In the case of the k-NN it is a 3-dimensional space so the coordinates have 3 points (x,y,z).
To calculate the coordinates of the centroid (G), the following equation is executed for each set of predicted positions:

$$x_G = \frac{x_1 + x_2 + \ldots + x_n}{n} \qquad y_G = \frac{y_1 + y_2 + \ldots + y_n}{n} \qquad z_G = \frac{z_1 + z_2 + \ldots + z_n}{n}$$

Figure 4.4: Centroid equation

Chapter 5

Experimental results and discussion

This chapter exposes the results obtained and gives interpretation and discussion on these.

5.1 Datasets comparison

As previously explained, the k-NN algorithm was initially intended to be trained on seven distinct datasets. However, due to the lack of time, only five datasets were used: DSI1, DSI2, LIB1, LIB2 and MAN2. MAN1 and SIM001 are the largest ones and the model takes very long time to execute, around 2 hours depending on the parameters. This is actually one of the weak points of this algorithm, the computation time can be very high for large datasets with many samples.

The hyperparameters involved to test the accuracy are the k value, the distance metric, and the data representation (normalization, absolute value and increment).

To evaluate the algorithm and compare the results with the different datasets, the values of k selected for the training are k=1, k=5 and k=11. The following table shows the average error for each dataset using the Euclidean distance metric:

	Average error		
Dataset	**k=1**	**k=5**	**k=11**
DSI1	6.46	6.13	**5.38**
DSI2	6.46	5.26	**5.04**
LIB1	3.8	3.17	**3.03**
LIB2	4.47	4.07	**3.95**
MAN2	**3.26**	3.27	3.4

Table 5.1: DSI1 average error with Euclidean distance

It can be observed that the first four datasets reduce their average error when the value of k increases, however MAN2 is the only one that gets worse and the average error is higher at each iteration.

This fact occurs because of the density of the datasets. The density of MAN2 is lower and a high k value implies that the algorithm selects fingerprints very distant from the test sample and the estimated position obtained can be biased.

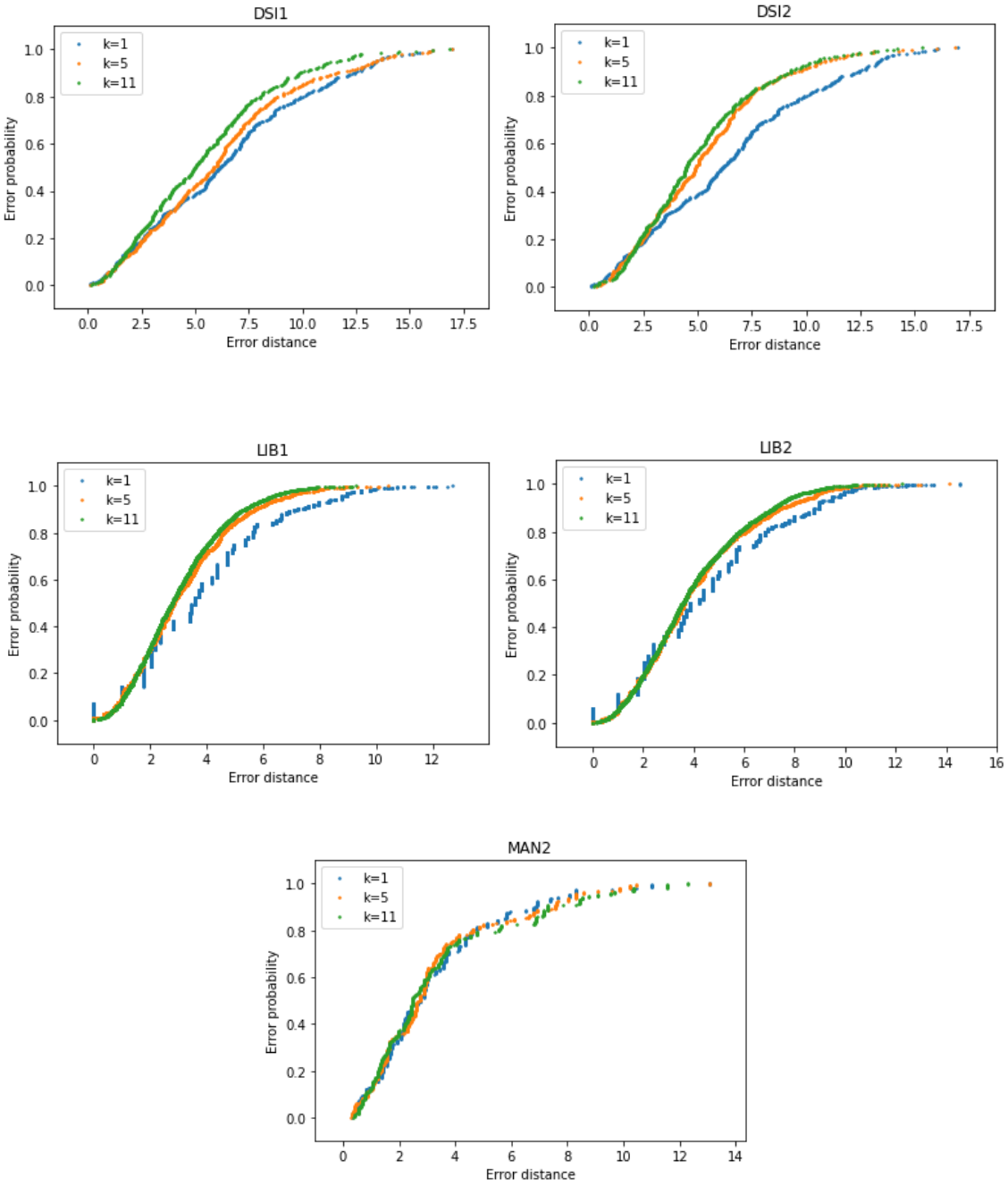

Figure 5.1: ECDF plots for each dataset indicating the error distance

The ECDF plots show the error distance in the x-axis and the probability in the y-axis. The more “vertical” the line is, the better are the results because it means the average error is lower. For the first four datasets, the green line (k=11) is on top of the others, indicating a better performance as shown in the table 5.1.

5.2 Distance metrics comparison

The comparison of the distance metrics implies the fingerprints transformation to avoid the errors mentioned in the previous chapter. The first experiments show the results obtained on the DSI1 dataset for k=1 and the three data representations:

	k=1		
Distance metric	**Normalization**	**Abs value**	**Increment**
Euclidean	6,52	5,04	6,46
MinkowskiL3	6,34	5,58	6,38
MinkowskiL4	6,52	6,25	6,46
MinkowskiL5	6,59	6,62	6,56
City Block	6,49	4,95	6,47
Chebyshev	6,48	7,67	6,53
Sorensen	6,34	26,65	6,38
Gower	21,76	4,95	21,18
Soergel	25,76	4,99	26,65
Kulczynski d	6,34	4,99	6,38
Lorentzian	6,28	5,66	6,28
Canberra	6,28	4,8	6,28
Intersection	6,18	4,95	5,91
Wave Hedges	5,82	4,7	5,81
Czekanowski s	6,34	4,99	6,38
Czekanowski d	5,83	51,34	5,77
Motyka s	6,28	4,99	6,28
Motyka d	46,62	51,34	47,32
Kulczynski s	6,28	4,99	6,28
Ruzicka	46,62	4,99	47,32
Tanimoto	6,28	4,99	6,28
Inner Product	6,28	40,68	6,28
Harmonic	6,28	57,31	6,28
Cosine	39,32	5,06	39,76
Kumar-Hassebrook	28,16	5,1	28,73
Jaccard s	6,43	5,1	6,42
Jaccard d	6,73	46,01	6,51
Dice s	6,73	5,1	6,51
Dice d	39,25	46,01	47,16
Fidelity	6,47	40,68	6,51
Bhattacharrya	45,94	57,31	47,16
Hellinger	40,14	5,01	40,56
Matusita	40,53	5,01	41,54
Squared-chord	6,44	5,01	6,29
Squared Euclidean	6,44	5,04	6,29
Pearson	6,44	5,11	6,29
Neyman	6,52	4,87	6,46
Squared	16,38	5,01	13,91
Probabilistic Symmmetric	13,51	5,01	8,75
Divergence	6,47	4,77	6,45
Clark	6,47	4,77	6,45

Additive symmetric	6,36	4,99	6,09
Kullback Leiber	6,36	57,31	6,09
Jeffreys	11,81	5	7,34
K Divergence	24,71	57,31	24,84
Topsoe	6,94	5,01	6,52
Jensen-Shannon	39,94	5,01	42,63
Jensen difference	6,46	5,01	6,39
Taneja	6,46	5	6,39
Kumar Johnson	6,46	4,98	6,39
AvgL	7,33	4,97	6,61
Vicis-Wave Hedges	15,08	4,76	8,35
Vicis Symmetric 1	6,35	4,9	6,41
Vicis Symmetric 2	12,87	4,91	7,18
Vicis Symmetric 3	15,49	5,02	10,25
Min-Symmetric	12,98	4,98	7,24
Max-Symmetric	6,27	5,02	6,43
Mean value	**13,71**	**13,48**	**13,33**

Table 5.2: DSI1 distance metrics comparison for k=1

The results obtained are all very similar but as the best average error appear for the Increment transformation, the next experiment consists of test the model using this data representation and different k values:

Distance metric	**k=1**	**k=5**	**k=11**
Euclidean	6,46	6,13	5,38
MinkowskiL3	6,38	6,27	5,42
MinkowskiL4	6,46	6,18	5,52
MinkowskiL5	6,56	6,23	5,62
City Block	6,47	5,95	5,11
Chebyshev	6,53	22,08	24,39
Sorensen	6,38	24,98	24,31
Gower	21,18	5,95	5,11
Soergel	26,65	5,86	5,12
Kulczynski d	6,38	5,86	5,12
Lorentzian	6,28	5,55	4,84
Canberra	6,28	5,3	4,74
Intersection	5,91	5,95	5,11
Wave Hedges	5,81	5,37	4,75
Czekanowski s	6,38	5,86	5,12
Czekanowski d	5,77	45,89	43,95
Motyka s	6,28	5,86	5,12
Motyka d	47,32	45,89	43,95
Kulczynski s	6,28	5,86	5,12
Ruzicka	47,32	5,86	5,12
Tanimoto	6,28	5,86	5,12
Inner Product	6,28	39,65	36,56
Harmonic	6,28	25,41	24,86

Cosine	39,76	6,02	5,35
Kumar-Hassebrook	28,73	6,04	5,34
Jaccard s	6,42	6,04	5,34
Jaccard d	6,51	45,71	43,66
Dice s	6,51	6,04	5,34
Dice d	47,16	45,71	43,67
Fidelity	6,51	40,84	36,24
Bhattacharrya	47,16	34,07	33,37
Hellinger	40,56	6,05	5,39
Matusita	41,54	6,05	5,39
Squared-chord	6,29	6,05	5,39
Squared Euclidean	6,29	6,13	5,38
Pearson	6,29	14,02	13,34
Neyman	6,46	8,27	7,42
Squared	13,91	6,14	5,35
Probabilistic Symmmetric	8,75	6,14	5,35
Divergence	6,45	5,77	5,17
Clark	6,45	5,77	5,17
Additive symmetric	6,09	7,04	6,55
Kullback Leiber	6,09	23,29	23,42
Jeffreys	7,34	6,13	5,59
K Divergence	24,84	34,99	33,47
Topsoe	6,52	6,07	5,36
Jensen-Shannon	42,63	6,07	5,36
Jensen difference	6,39	6,07	5,36
Taneja	6,39	6,25	5,68
Kumar Johnson	6,39	8,23	7,79
AvgL	6,61	5,95	5,11
Vicis-Wave Hedges	8,35	6,88	6,55
Vicis Symmetric 1	6,41	9,97	9,73
Vicis Symmetric 2	7,18	7,18	6,76
Vicis Symmetric 3	10,25	6,03	5,26
Min-Symmetric	7,24	12,65	11,44
Max-Symmetric	6,43	7,53	6,72
Mean value	**13,33**	**12,76**	**11,95**

Table 5.3: DSI1 distance metrics and k value comparison

From the results above, the lowest average error is for k=11, which matches with the results obtained on the section 5.1. It definitely indicates that DSI1 dataset requires a higher value of k. The next step is to select three of the best distance metrics to compare the results using an ECDF plot. The metrics selected for comparison should be from different families to enrich the experiment. In this case, the ones selected are Euclidean, Gower and Wave-Hedges:

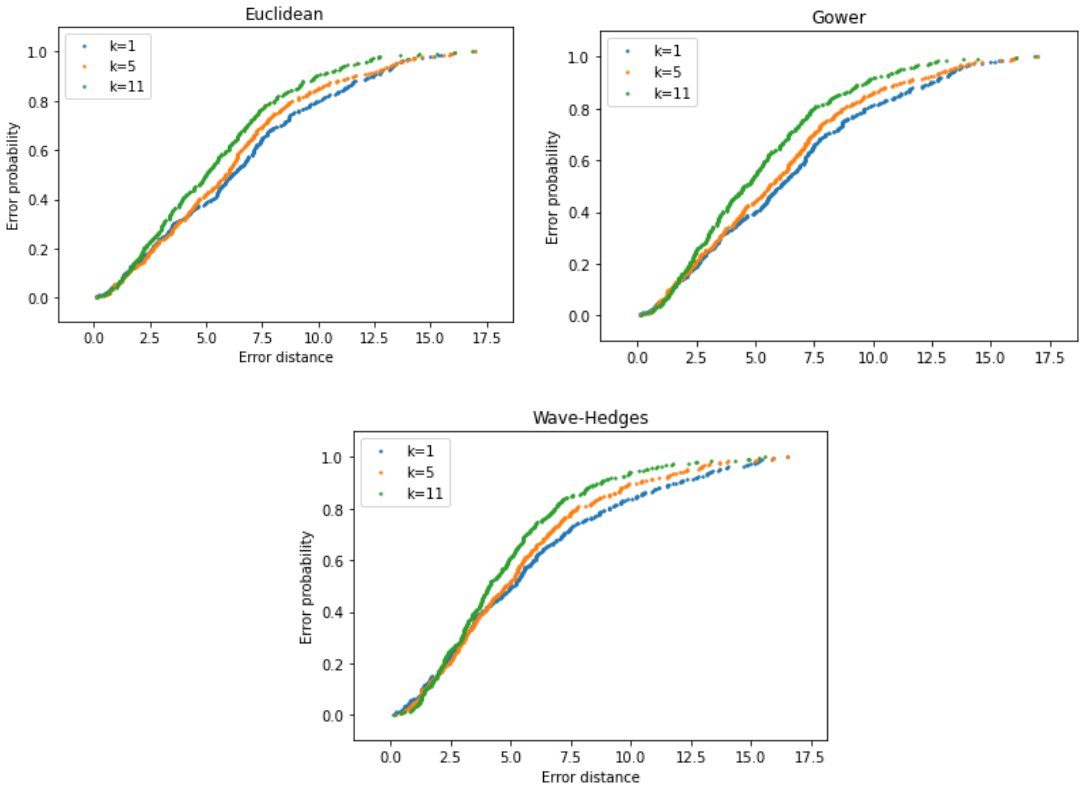

Figure 5.2: DSI1 three metrics ECDF comparison

Wave-Hedges shows the most vertical lines, specially for k=11. It is the one with the lowest average error of them: 4,75 meters. The points in the higher part of the plot are more dispersed, which indicates that the high error probability corresponds to fewer samples.

5.4 Ensemble

Considering the signal propagation and the fact that the k-NN process is not totally deterministic but has some part of randomness, the final proposal is to make an *ensemble*.

The next step to improve the average error is to make an ensemble of the three previously selected distance metrics and evaluate the results. The following scatter plots show the comparison between the positioning error of the ensemble against the positioning error of the metric alone:

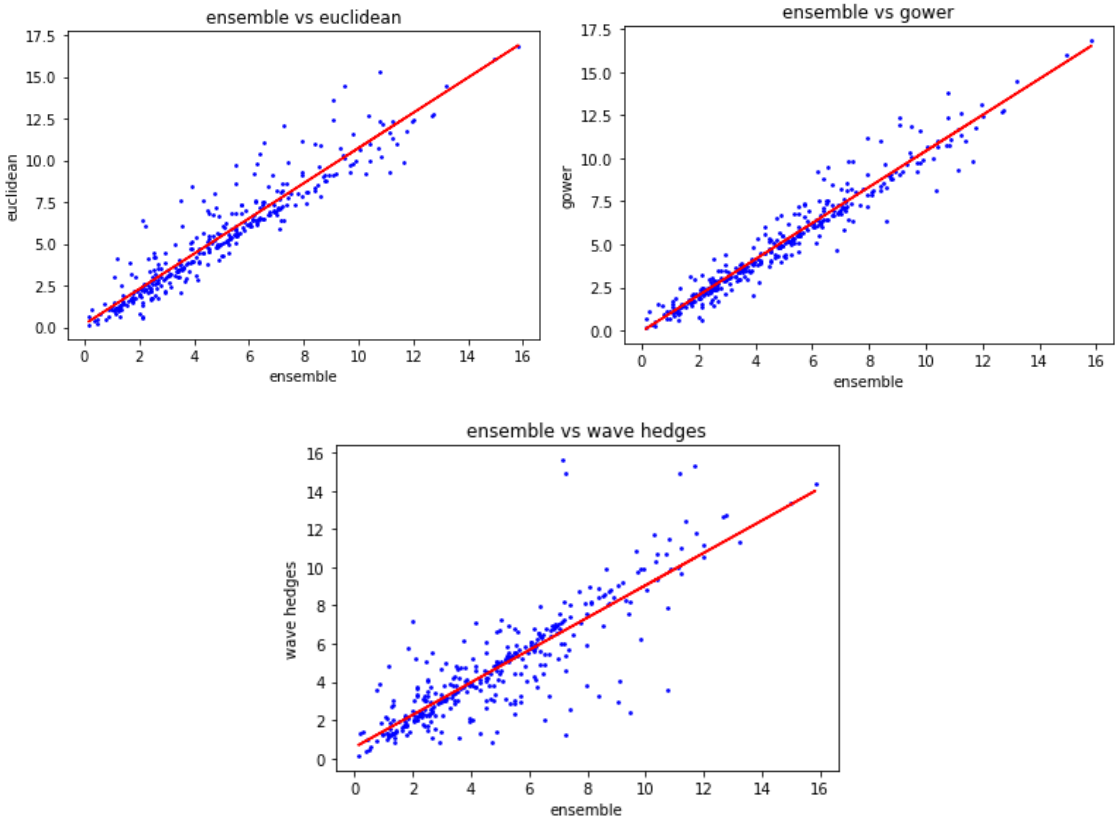

Figure 5.3: DSI1 scatter plots for three distance metrics ensembled

If the points on the scatter plot are very close to the regression line it means the error on both methods is very similar but, in this case, we can see a remarkable dispersion. The points over the line indicate a better performance of the ensemble and the ones under the line indicate better performance of the single metric.

Also an ECDF plot of the previous variables can be created:

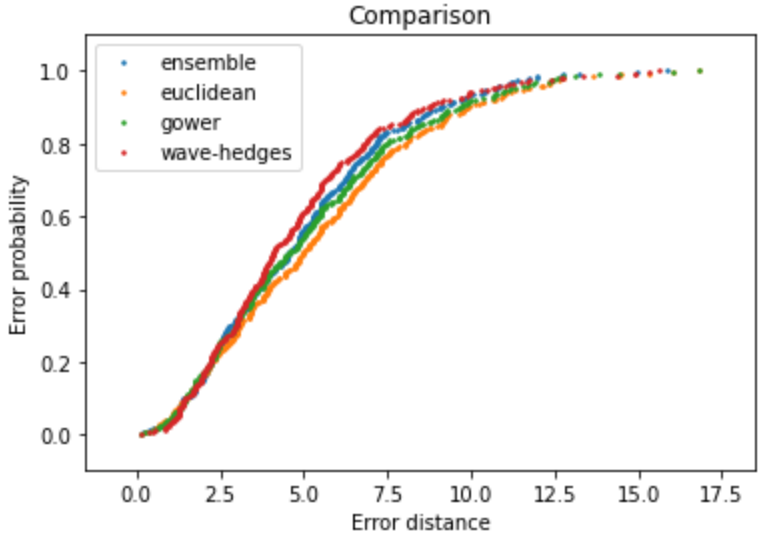

Figure 5.4: DSI1 ECDF plot for three distance metrics ensembled

Wave-Hedges turns out to be the best performing metric, even better than the ensemble. The numeric results of the previous calculations are as follows:

Metric	**Average error**
Ensemble	4.92
Euclidean	5.38
Gower	5.11
Wave-Hedges	**4.75**

Table 5.4: DSI1 average error ensemble and selected metrics

The ensemble achieves a good average error but not the best. It is a normal behaviour of this process since it combines the results and outputs the mean value by calculating the centroid.

5.5 Dataset and distance metrics comparison

The final experimentation to evaluate the accuracy of the model in the rest of the datasets is to create an ensemble with the same parameters of the previous section for DSI2, LIB1, LIB2 and MAN2:

Dataset	**Average error**
DSI1	4.92
DSI2	4.79
LIB1	2.87
LIB2	3.96
MAN2	10.86

Table 5.5: Average error of ensemble for all datasets

Since the components of the ensemble have been chosen for DSI1, the results obtained are disparate. It is an indication to see that the influence of the dataset characteristics is very high on this algorithm. The following plot shows the ECDF of the previous results:

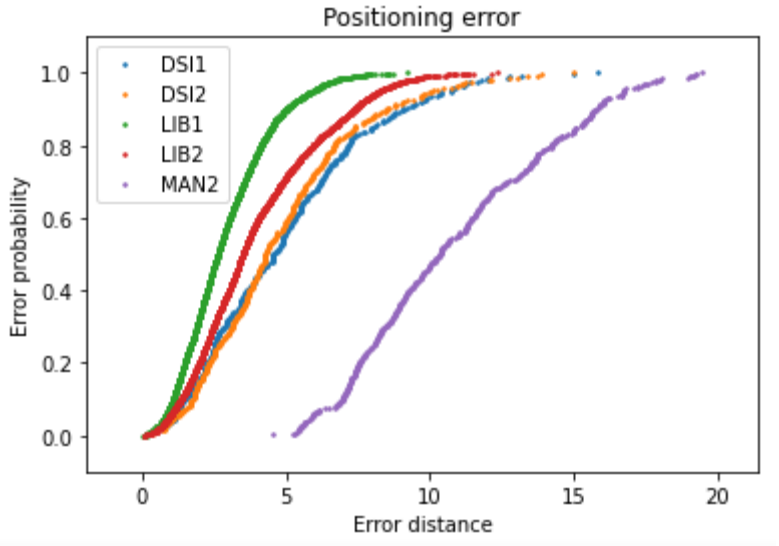

Figure 5.5: ECDF plot of ensemble for all datasets

Chapter 6

Conclusions and future works

6.1 Conclusions

The initial objective of this work was to implement an algorithm based on the k-NN to perform geolocation in indoor environments minimizing the positioning error. After all the tests explained in the previous episodes, the main conclusion would be that it is complicated to find “the best” distance metric or the best configuration to perform a k-NN execution. Probably the most influential variables to consider is the difference between the datasets since it plays a key role on this problem: density, size, etc.

One of the difficulties encountered during the development of the project was the long execution times of the algorithm for some datasets, particularly MAN1 and SIM001. Initially, the idea was to train the model with seven different datasets to compare results but that was finally discarded because of the lack of time. MAN1 and SIM001 included too many fingerprints and the execution of all distance metrics could last up to two or three hours. This is in fact one of the main disadvantages of the k-NN algorithm.

The experimentations done during the development of the project are the following:

- Datasets comparison: overall results varying the k value and using Euclidean distance.
- Distance metrics comparison: evaluate the model with all distance metrics and using the three proposed data representations.
- Ensemble: select the best metrics and calculate the ensemble to achieve the best results.

The general verdict of this study is that the *ensemble* is the best option to achieve a good overall performance. After testing the model with several metrics, an ensemble of the top ones ensures a robust result.

6.2 Future works

The results presented still leave much scope for improvement. Future works to improve the performance are suggested:

- Distance metrics Python implementation is made using *numpy* library to facilitate de coding. However, it slows down considerably the execution times, a faster way could be found for that purpose.
- The normalization of the fingerprints is done separately for train and test datasets. Some results are rounded to 0 as explained in section 4.2 and it causes a slight modification of the data.
- Repeat all the analysis considering the floor and building to get a more complete information of the position.

Chapter 7

Bibliography

[] https://drive.google.com/drive/folders/1IRLcwe2jnZqer6y1nuMuAooReKT25o_J

[] PENDING