# PAC 3 – Design and implementation

1. Introduction

2. Related work

3. Materials and Methods

4. Description of the proposed approach

5. Experimental results and discussion

6. Conclusions

Functions:

1. Get\_neighbors() -> euclidean distance default. Calculate distance between each training rss and each test rss. Return list of k indexes of nearest training rss for each test row.
2. Knn() -> iterate test rss and find list of ids of k nearest neighbours (train data)
3. Predict\_position() -> predicted coordinates of previous knn. Iterate previous list of ids and calculate mean value of coordinates (x,y,z) of train coordinates found by index. Return list of calculated coordinates.
4. Calculate\_error() -> euclidean distance of previous predicted positions and actual positions. Return list of error distances between previous prediction and test coordinates.

Plots:

* ECDF: sort error distances and plots with error probability.

Average error: calculate mean of error from list (output of calculate\_error function)

Comparison of Euclidean distance implementation: since it is the shortest distance and most common and also used to calculate error between prediction and reality, I compared several implementations to find the most efficient:

* “handmade” with math.sqrt
* linalg.norm from numpy
* square, sum and sqrt from numpy
* distance.euclidean from distance package
* math.dist from math package

Old exec times before modifying get\_neighbors function:

Texto

Descripción generada automáticamente

Distance metrics comparison: implementation of distance metrics using public packages or “handmade” implementation using numpy.

Errors encountered:

Texto

Descripción generada automáticamente

Solutions:

1. Abs value of rss
2. Increment rss by constant value (100)
3. MinMax normalization:
   * First attempt: minmax of every RSS -> incorrect because scale is lost
   * Second attempt: minmax of trnrss and tstrss -> incorrect because scale is lost again (but less than before)
   * Third attempt: scale train data and transform train and test with the same scale to keep same min and max limits.
4. Add constant (1e-7) to P and Q before division and log (or always?).
5. Some metrics don’t show error but the avg error looks incorrect because is the value that always appears when error occurs (min\_symm, max\_symm and sometimes Hellinger and Matusita).

Notes:

* Work with numpy arrays facilitates computation but is much slower