ADVANCED PROGRAMMING
MASTER IN STATISTICS FOR DATA SCIENCE.
RCPP ASSIGNMENT: PROGRAMMING NEAREST NEIGHBOUR IN C++
3.0 POINTS

Introduction

The aim of this assignment is to program in C++ a simple but useful machine learning method: KNN or k-nearest neighbor. I have already programmed KNN with K=1 in R, **for classification**, which you will find in Aula Global. Steps to follow in the assignment. Use different function names for each of the steps.

1. **(0.5 points)** Write function **my_knn_c**, to translate my R code into C++
    - Write comments in your C++ to show that you are undersanding what the code does.
    - In this case, compile the C++ code with *sourceCpp*. Show in your report that you have used *sourceCpp*.
    - Check that you get the same results than the knn that belongs to library *FNN* or library *class* (choose one)
    - Use the library microbenchmark in order to determine whether the C++ version is faster than:
        - the R version of the code
        - and the knn of *FNN* / *classif*
2. **(0.5 points)** Define a new C++ function for the Euclidean distance between two points and use it in your code. Use as much sugar as possible for the function. This function will be called from your main knn function, that will be called **my_knn_c_euclidean**.
    - From this point onwards, put every function in **your own package** (as explained in the lecture). Use different function names for different steps in the assignment.
3. **(0.5 points)** Define now a function for the Minkowsky distance (instead of Euclidean) and use it in your code. Add a new argument for exponent p to your main knn function (**my_knn_c_minkowsky**). If p positive, then use p as exponent. If negative, then it means L_∞
4. **(1.5 points)** Finally, write a function **my_knn_c_tuningp** that instead of using the exponent p that is passed as an argument to do hyper-parameter tuning of the Minkowsky exponent p. In order to do so, add a new argument to your main function with the vector of integer exponents to try (*possible*_p)
    - Split the training data X into 2/3 for train-train and 1/3 for train-validation.
    - Go through all values in *possible_p* and compute the accuracy of knn on the train-validation partition for each possible p
    - Finally, use the best p for making the prediction for X0

What to hand in:

- Hand in your code and report in Aula Global. It should be a zip file containing:
    a. The report. It should give a brief description of what your functions do and show that your functions do what they are meant to do
    b. A source R **package** (as taught in class) with all the functions (a different name must be used for each function).