CSCI 611 - LAB#3 SPECIFICATIONS

OBJECTIVE:

- Practice python coding (`NumPy` arrays, `math` module, functions)
- Understand kNN algorithm and build model from scratch (`Python`)
- Visually Explore available/approved datasets
- Assess impact of varying specific hyper-parameters

PREPARATION & REFERENCES:

- IMLP (Muller/Guido book) Ch.2.3.2
- `Sklearn.datasets`
- SciPy Lecture Notes 1.4 (NumPy arrays)
- Matplotlib.pyplot

TASK 1 - IMPLEMENTATION

Design & implement functions necessary to support creation of kNN estimators. You are required to build the functionality from scratch, in Python. You may *not* rely on kNN functionality provided through the SciKit learn (or other ML) API. However, you *may* incorporate basic python array functionality from `numpy`.

Your solution must adhere to the following requirements:

1. Your solution must be **your code**, and implemented in Python.
2. Any imported libraries or modules beyond the following are restricted (unless you have a demonstrated need, legitimate request, and approved in writing):
   a. `numpy`
   b. `matplotlib`
   c. `sklearn.datasets`
   d. `math and statistics` - use minimally (you may *not* use built-in *dist* or *sort* functions)
3. Your solution should allow me to custom build kNN models, fit them with different datasets, and predict on new datapoints. The following functions should be invocable directly (and in unison) from the command line, OR, **if** you choose to design a **knn class** then ***knnBuild*** should create an object, with which to invoke the ***knnFit*** and ***knnPredict*** as methods. Be sure to provide documentation and a simple runtime example. The following **HIGH LEVEL functions/methods** must be explicitly supported as indicated (by name and parameter name/type/order).
   a. ***knnBuild***(model, k, algorithm="euclidean",p=2)
      model is {"classifier","regressor"} - required
      k is int(number of neighbors) - required
      algorithm is {"euclidean","manhattan","minkowski"} – default is "euclidean"
      p is int(power for distance formula) – default is 2
   b. ***knnFit***(X,y) – invoked after *knnBuild*, as a function or method
      X is feature data for all data-points, as a 2D numpy ndarray
      y is the target/label for each data-point, as a 1D numpy ndarray
   c. ***knnPredict***(xnew,…) – invoked after *knnFit*, as a function or method; this is when the work begins! It is inevitable that invoking this function/method sets off a chain of function calls, some of which must include those mentioned next. Include any additional parameters in the function header that you need if you implement it as a function rather than method. Document clearly what those parameters are in type, order, and semantics.
4. Your solution must also explicitly support the following **HELPER functions** & parameters. I should be able to invoke any of these directly from the command line. These functions must be supported as indicated (by name and parameter name/type/order), and not as class methods.
   a. ***quicksort***(arr,int) – receives a 2D numpy ndarray (arr), sorts (asc) by specified column (int), and returns it
   b. ***euclidean*** (arr1, arr2) – returns the *euclidean* distance between two 1D numpy ndarrays
   c. ***manhattan***(arr1, arr2) – returns the *manhattan* distance between two 1D numpy ndarrays

d. ***minkowski*** (arr1, arr2, p=2) – returns the *minkowski* distance between two 1D numpy arrays, power is an int, set to default value

5. Additional low level **SUPPORT functions/methods** may be implemented, as needed, to support the basic functionality of your high level functions. You may, but are *not* required to implement a kNN class.

6. Your solution must be limited to the requested functionality & features … in other words, it should not be unnecessarily complex or bloated. Furthermore, the code must be documented & clean of troubleshooting when submitted.

7. You *MAY* earn bonus points (up to 10%) for including 1 additional FEATURE. The feature must be in the form of an additional method or function of your choosing. It should provide new functionality (not a replacement functionality). It must be documented.

TASK 2 – TESTING - choose sample data from scikitlearn, mglearn, or other approved sources

1. TEST 1 – kNN CLASSIFIER
    a. Load sample classification data
    b. Visualize the data
    c. Test your custom kNN classifier on the data
2. TEST 2 – kNN REGRESSOR
    a. Load sample regression data
    b. Visualize the data
    c. Test your custom kNN regressor on the data
3. Prepare TEST 1 and TEST2 as 2 separate notebooks for submission:
    a. Remove any unnecessary cells
    b. Document test run with comments as text cells
    c. Run & Expand to show both code & outputs
    d. Save as .ipynb AND pdf
4. You should of course compare your results to *scikit learn* results, under similar circumstances. Don't expect exact results, but if yours are far off you should investigate further and explore your code for errors. Make not of these findings, but DO NOT submit this runtime comparison (see NARRATIVE below).

TASK 3 - NARRATIVE:

1. General header to include your name AND that of any classmates you brainstormed with (including in-class breakout session(s).
2. 1 brief paragraph describing your BONUS FEATURE (if you have one), and how to invoke it.
3. 1 brief paragraph describing the performance of your implementation to that of *scikit learn* knn.
4. 1 brief paragraph reflecting on how you would improve your implementation for future upgrades.

STYLE – for all file submissions

- Include a general header (file name, your name, course, date, brief description)
- Include comments/text cells generously, to document your code
- Place ALL IMPORT STATEMENTS in a single cell directly after your general header. You will receive significant point reductions for any import statements located within body of your solution.

DELIVERABLES

- ***xy***Lab3knn.py – TASK1 - Python source file of kNN implementation, where ***xy*** are your initials;
- ***xy***Lab3test1.ipynb and .pdf – Your TASK2 - TEST1 demonstration.
- ***xy***Lab3test1.ipynb and .pdf – Your TASK2 – TEST2 demonstration
- ***xy***Lab3narrative.pdf – Your TASK3 Narrative