E4 spec sheet:

Normalized error

Mission command preamble: As in general, we won't tell you *how* to do something. That is up to you and your creative problem solving skills. However, we will tell you what we would like this function to do. So here are the specs of the function we would like you to write.

Purpose: In class, you have learned about the most common metric to gauge the accuracy of machine learning methods – RMSE (root mean squared error). However, just like with vector norms, RMSE is not the only way to measure the error of prediction algorithms. For instance, if one is dealing with ordinal data or data with lots of outliers one would be well advised to calculate the mean absolute error instead. As this idea extends to cubed and higher order errors, we would like you to write a general normalized error function.

Specific things we would like this function to do:

a) Take in two arguments/inputs, in this order: 1) An input dataset (make it a 2D numpy array or dataframe), 2) a flag by which power to normalize the error, for instance 1 for the mean absolute error, 2 for the RMSE, 3 for the cubic root mean cubed error and so on.

b) Calculate the error of the input array in a1, normalized by the flag set in a2, according to the following general equation: $NE = \sqrt[p]{\frac{\sum(|\hat{Y}_i - Y_i|)^p}{n}}$ where $\hat{Y}$ are the predictions, Y are the measurements, p is the power specified in a2, n is the number of rows of the array in a1, and i is the i_{th} row, from 1 to n, yielding the normalized error NE.

c) The output of this calculation should be a single scalar, real number.

d) The function should return the number from c), once all calculations are done.

e) Assumptions: You can assume that the input will be a 2D array (with predictions in the column 1 and measurements in column 2). The number of rows of this input array should be flexible/up to the user.

f) Make sure the function has a clear header as to what inputs the function assumes, what outputs it produces and when it was written.

Input / output examples:

Input array A:

1	**2**
2	**4**
3	**1**
4	**6**
5	**2**

Input array B:

0	**50**
10	**5**
5	**10**

Output of normalizedError(A,1): 2

Output of normalizedError(A,2): 2.0976

Output of normalizedError(A,3): 2.1828

Output of normalizedError(B,1): 20

Output of normalizedError(B,2): 29.1548

Output of normalizedError(B,3): 34.6912