You are tasked to develop a regression model that best fits the data below. You will turn in a Jupyter notebook in both PDF format and also the notebook file, which describes your method and provides and demonstrates your code.

All data sets are curated from the UCI Machine Learning repository.

- Airfoil Self-Noise: https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise (Links to an external site.)
- Yacht Hydrodynamics: https://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics (Links to an external site.)
- Concrete Slump: https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test (Links to an external site.) (note that this data set has 3 output variables)

1) Develop a function with the following first line:

def my_regression(trainX, testX, noutputs)

The input variables are:

- trainX - an [ntrain x (nfeature + noutputs)] array that contains the features in the first 'nfeature' columns and the outputs in the last 'noutput' columns
- testX - an [ntest x nfeature] array of test data for which the predictions are made
- noutputs - the number of output columns in trainX

The output should be an [ntest x noutputs] array, which contains the prediction values for the testX data. You can use these data to then calculate squared error by comparing against the testX outputs.

Your my_regression code should do some kind of cross-validation to determine the right model for the training data, e.g., linear vs. polynomial vs. radial basis functions (your choice). Then this model is applied to the test data to make a prediction.

Rules:

- Your code should not use any regression library, but can use NumPy as this will be useful for array handling.
- No other inputs to this function are allowed.
- You may not use the testX data to train your model. Notice that outputs are not passed in for testing data.
- Your code can perform data scaling (scaling features to the interval [0,1] or z-score scaling), can use any basis functions you want, cross-validation on the training data, regularization, and model selection.

2) Test your regression on a simple one-dimensional data set, where your training data is generated by the following function:

$$y = \cos(x_2) + 0.1 x_3 + \epsilon, \qquad x \in [-5, 5]$$

where $\epsilon = N(0, \sigma_2)$ is additive white Gaussian noise with variance σ_2.

a) (5 points) Plot y versus x (y on vertical axis, x on horizontal axis) as a line with noise variance = 0. This is the truth that you wish your regression model to fit to. Plot from x = -5 to +5 at increments of 0.1.

b) (5 points) Plot y versus x for standard deviations $\sigma = 0.1, 0.2, 0.5, 1$. Each line should be a different color. Make sure your plots are labeled with a legend. You will now use your regression function to see how close you can come to the truth in part a) based on noisy measurements.

c) (10 points) Now you will run an experiment where you will see how close your regression is to the truth based on the variance of the noise and the number of training data. Make a table that has number of training data [2,5,10,20,50,100,200,500] in the rows and the standard deviations from part b) in the column. Run 5-fold cross validation on each of these 32 combinations of parameters and report the cross-validation squared error.

d) (10 points) Select the two best results (lowest error) and two worst results (highest error) from part c) and make four plots. Each plot should show the truth line you plotted in part a) and the learned regression model. Plot from x = -5 to 5 at increments of 0.1. What do you notice about these results?

3) Now run your regression algorithm for the three UCI data sets above. Run five-fold cross-validation to estimate the squared error of your learned model for each data set. Comment on the results.

Present your findings for each data set as a Jupyter notebook. Submit both a PDF and the notebook file. Present your squared error for several cross-validation folds of each data set.