**CSCE 623: Machine Learning**

**Spring 2019**

**HW3**

Due Tuesday, 30 April at 2359

Submit via Canvas

**(**This Homework is worth 5 points toward your final grade**)**

Your homework will be composed of an integrated code and report product using Jupyter Notebook. In your answers to written questions, even if the question asks for a single number or other form of short answer (such as yes/no or which is better: a or b) you must provide supporting information for your answer to obtain full credit. Use python to perform calculations or mathematical transformations, or generate graphs and figures or other evidence that explain how you determined the answer.

This homework explores cross-validation. You will be working with synthetic data. This homework is inspired by problem 8 from Chapter 5 in your text.

Each step listed below should correspond to python code and/or text in your report and code files, and the step number should be clearly indicated in both the code and your report.

**Instructor provided code:**

1. Two helper functions are provided below  
   The first generates datasets. You can call it in later code chunks. Note that in this data, *x* is a predictor / feature and *y* is a response variable.

def makeData(myseed=1, quantity = 100):

np.random.seed(myseed)

x = np.random.uniform(low=-2.,high=2.,size=quantity)

y = x - 2 \* (x \*\* 2) + np.random.normal(size=quantity)

df = pd.DataFrame({'x': x, 'y': y})

return(df)

This second helper function generates a polynomial design matrix from a single feature vector *x*. The returned matrix *X* contains columns of x\*\*0, x\*\*1, … x\*\*p where p is the desired highest order of the polynomial. Note that since it returns a design matrix, the columns correspond to *β*0 through *βp*

def polyDesignMatrix(x, p):

x = np.array(x)

X = np.transpose(np.vstack((x\*\*k for k in range(p+1))))

return(X)

**STUDENT CODE – include your code in the code cells following the step numbers in the ipython notebook**

**Make & explore the data:**

1. Call the function to make a dataset: df1=makeData(). Answer the following questions: How many observations? How many features/predictors? Determine and display the value of *n* (count the observations) and the value for *p* (count the predictors (features)) in this dataset
2. Create a scatterplot of X against Y. Describe the shape of the data. What kind of relationship will fit the data? Linear? Polynomial (and if so, what order of polynomial)? Form an official hypothesis about the best order model and state it in a markdown cell.

Implement OLS coefficient determination and prediction

1. Define two functions.

The first function computes the coefficients for ordinary least squares from a design matrix X and the response variable y. The signature for the function is getOLScoefficients(X, y). Note that the first column in a design matrix should be a column of ones in order to properly fit the intercept term.   
The second function computes the predictions (yhat) from a design matrix and a set of coefficients. The signature for this function is getOLSpredictions(X, betas). The function should return a column vector of predictions, one for each row in X

**Cross Validation (inspired by problem 8)**

LOOCV:

1. Define a function to run LOOCV to return cross-validation performance on an OLS regression model with polynomial terms. The signature of a call to this function is LOOCVerr(df, modelOrder), where the dataset is df and the maximum term order is defined by modelOrder. This function should return a vector of *n* cross validation error values (squared error terms) that result from *n* repetitions of training the model on all but the *i*th observation and predicting on the *i*th observation.   
   For example, if modelOrder = 3, then your function will first obtain a design matrix *X* produced by polyDesignMatrix on the data feature x (*n* rows by 4 columns), and then run LOOCV on an OLS regression model for *y*=*β*0+*β*1*x*+*β*2*x*2+*β*3*x*3 using the X & Y data from df. Since df contains *n* observations then LOOCVerr will return a vector of length *n* containing the *n* individual squared error terms (actual *y* minus predicted *y*)2 .  
   **The goal of this step is for *you* to write code which manages the cross validation**. Call the functions to fit OLS coefficients and make predictions you wrote earlier from *within* LOOCVerr, and **write your own LOOCV cross-validation code to produce your results**.
2. Using df1 (where you ran makeData with a default seed value of 1) build a for-loop to run LOOCV to generate error vectors using modelOrder values from 1 through 4. LOOCV will build and return squared error vectors for 4 separate models which were evaluated with linear, linear+quadratic, linear+quadratic+cubic, and linear+quadratic+cubic+quartic terms.
3. Compute the MSEs from the error vectors and plot the MSE results from your LOOCV on models of order 1 through 4. This plot should have the model order on the x axis and mean squared error on the y axis (MSE is the mean of the *squared values of the error terms* on the y axis). Determine the model order with the minimum cross-validation MSE and indicate the minimizing model order on the plot & report it, along with the MSE for that model. Indicate whether or not the best order model matched your hypothesis in Step 2 and explain any differences.

Other Validation Methods:

1. Build another function to perform validation using the “validation set approach” described in ISLR section 5.1.1 where a randomly-selected half of the dataset is used for training, and the remaining portion is used for validation. Your function should have the signature VALSETerr(df,modelOrder,splitseed) and it should return a SINGLE MSE value of the prediction quality on the validation set. The randomness should be repeatable, based on controlling the random seed in the data permutation before the split using splitseed. When determining “half”, don’t forget to handle situations where the number of observations in df is odd.
2. Build another function to perform *k*-fold cross validation as described in the book section 5.1.3. This function will have the signature KFOLDerr(df,modelOrder,k,splitseed) and will return a *k*-length vector of total-error terms. Each total-error term represents the mean of the MSEs computed on each of the *k* folds. Membership of the data in each fold should be determined randomly. Hint: When partitioning the data into folds be careful to write code that handles non-integer fold-sizes appropriately (when the number of observations in df is not integer-divisible by k). The randomness should be repeatable, based on controlling the random seed in the data permutation before the determination of the fold memberships using splitseed.
3. In a later step you will visualize the *reliability* of 3 validation methods: A) validation set; B) 5-fold cross-validation; C) 10-fold cross-validation. Write code to compute and store the MSEs of each of the 3 validation methods (A, B, C) for each model order (1 through 4) on splitseed values of 1 through 10. You are collecting a total of   
   3 x 4 x 10 = 120 MSE values in this step (40 per validation method).
4. Make 3 “spaghetti plots” – one for each validation method (validation set, 5-fold CV, and 10-fold CV). In these plots, the X axis is model order and the Y axis is MSE. In each spaghetti plot there will be 10 lines (1 line per random seed which controlled the data split into train/val partitions). Each of the 10 seed lines will have 4 model order points which display the MSE at each of those model orders. For each line in a spaghetti plot, annotate the point with the lowest MSE using a datamarker (there will be one point indicated on each line).
5. Using your eyes and the plots from step 10, decide which of the validation techniques (validation set, 5-fold, and 10-fold) is **most reliable** for choosing model order on this dataset, and discuss your answer & reasoning.
6. Implement **code** for determining the overall **best-order** model from whichever was the most reliable validation method you selected in Step 11 (validation set, 5-fold CV or 10-fold CV). Report the best polynomial-order model chosen (1, 2, 3, or 4), indicate whether or not it matched your hypothesis in Step 2 and explain any differences.

**Rules of Engagement for this Homework Assignment:**

**Using external sources:**

The use of pre-existing solutions to answer assignments is not allowed. This includes the use of other students’ answers, answers found on the internet, solution manuals, and any other source of information which does not reflect your own work.

You may use the internet or get help from peers when determining basic things like “how do I add points to a plot in python” or how do I use sklearn, but don’t try to search for specific answers to problems I ask in the homework.

You may use any pseudocode or concepts learned in class to solve the problem.

The code you write must be original work.

**Submission Contents:**

You will submit a python notebook (jupyter), which contains markdown text, code and output/results in a single file.

**Programming Conventions**

In code, good software engineering principles apply: self-documenting code (meaningful function & variable names), additional comments and whitespace should be standard in all code you turn in.

Explain what you are doing in text in the markdown as well as in the comments within code cells. A rule of thumb is to have line-level comments in the code cells and save the larger high-level comments/discussion for the markdown text.

If datasets are provided by the instructor, place the dataset files in the same directory with your python notebook, and ensure that your python code loads and processes these files – your instructor will set up the same file structure when evaluating your code.

Do not hardcode a path on your computer to get to load data files – the instructor will not be using the same path you placed them in. Do not edit the datafiles – the instructor will use the original file when running your code for evaluation.

**Pre-submission Checklist:**

Ensure your text, code, and figures are present in the jupyter notebook. Do not submit the datafiles.

Ensure that each step in the homework description is clearly indicated in your python notebook. Your notebook steps should match the order of the steps indicated in the assignment. If you deviate from the order, include a markdown text description of the deviation and the rationale so your instructor knows where to look for your answer

Before submitting, save the file, close the jupyter window, shutdown the kernel, and then reopen the file and restart it to make sure you have a clean environment.

Once you have a clean environment, make sure you **run all** cells in from the beginning, and read through the output carefully to ensure your final product reflects what you intend to submit. Your instructor will rerun your code in a clean environment to make sure it works.

Make sure your name and date is in the markdown text at the top of the output document.

**Naming Conventions**

Your homework file name should be: “LASTNAME\_HW3.ipynb” where LASTNAME is your last name (use “LASTNAME\_FIRSTNAME\_HW3.ipynb” if there are two or more people in the class with the same last name).

**How to Submit**

Submit your zip file to Canvas.

**Resubmissions (error correction)**

Note that if you discover an error before the due date and change a problem solution and re-submit, keep in mind that your instructor will only review your latest submission on Canvas – make sure it is complete.