LAB 5 – Validation Comparison

In this lab, we will see the difference between cross-validation and validation. To do this, we will need the .csv provided with the assignment on Blackboard. Once again, our regression task stays the same: To investigate the effects of age, experience and power (x_1, x_2 and x_3) to the player's salary (y). You will be able to use any library you choose for this lab (hurrah!), although I don't recommend it since it will make things harder.

Instructions:

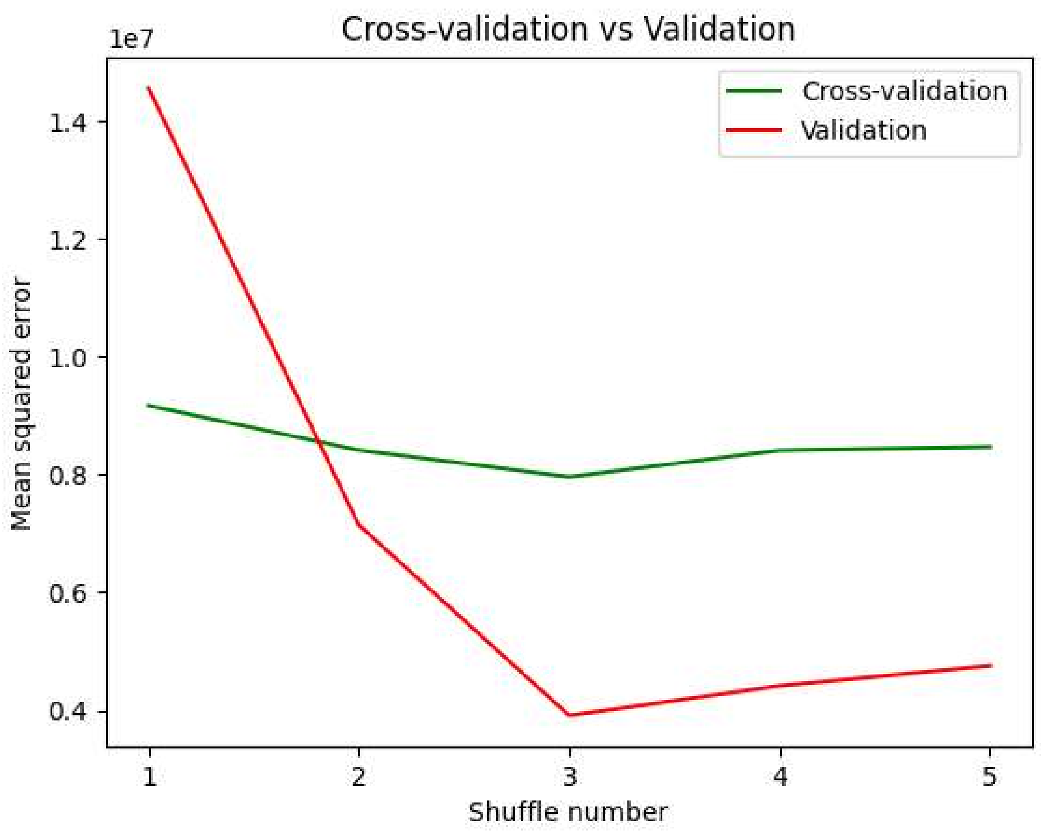
- (20 pts) Implement a function named `k-fold-cv`. This function should take in X (input matrix), y (output vector) and k as parameters and return the cross-validation error (in the form of mean squared error). This is the same task from the previous lab, therefore you can simply copy your code from the previous week, and put it inside a function, although small changes may be necessary.

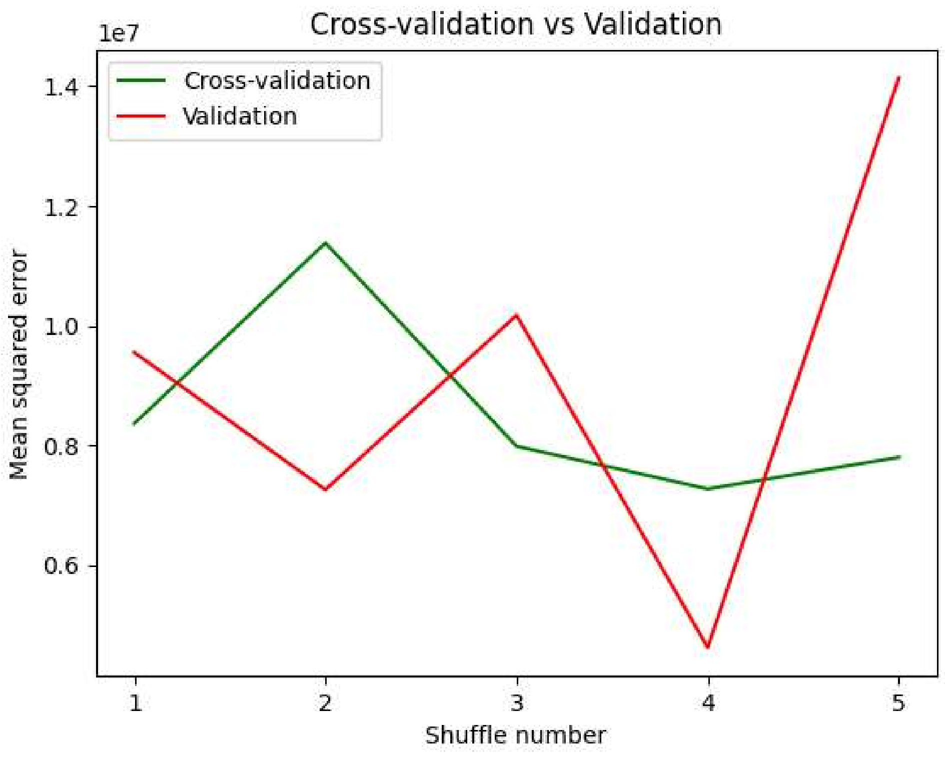
- (25 pts) Implement another function named `validation`. This function should take in X (input matrix) and y (output vector) as parameters. It should take the first %20 of the data as test data and the remaining %80 as train data. It should perform linear regression using train data and return the MSE using test data.

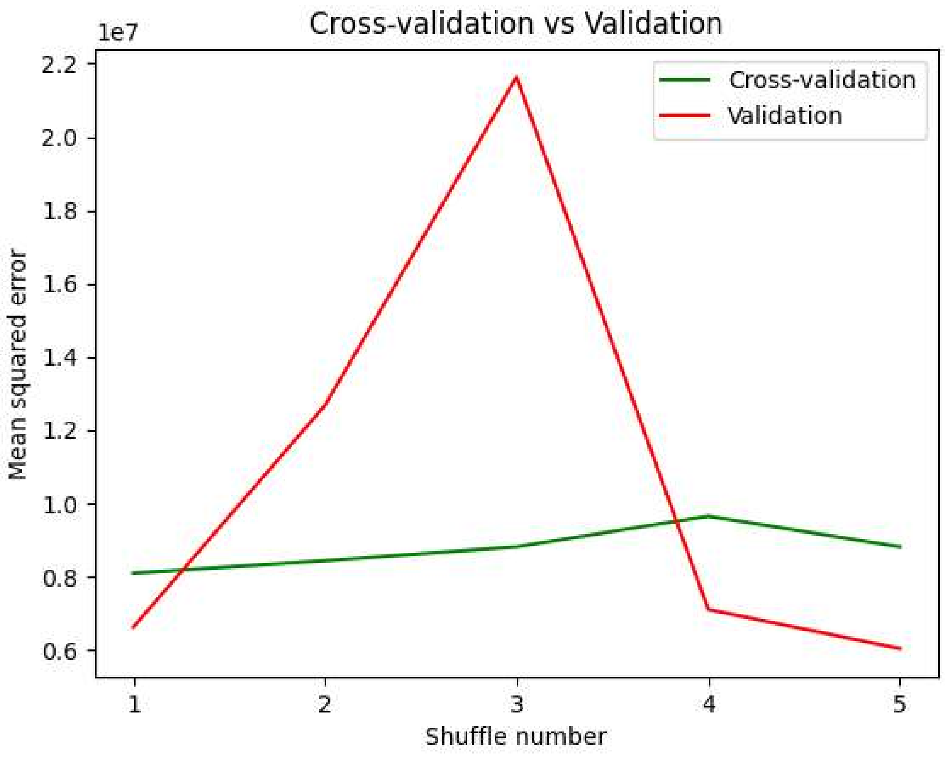
- (45 pts) In the main script, read the .csv file, extract columns, and form X and y. Then do the following 5 times:
  - Shuffle the data. This means randomly swapping rows of X and elements of y. But be careful, X and y should still correspond to each other (e.g. if row 1 of X is moved to row 5, element 1 of y should move to the 5th position as well)!
    An easy way to achieve this would be to first *combine* X and y beforehand, then shuffle (`numpy.random.shuffle()` is very useful for this), and then separate X and y.
  - Call `k-fold-cv` with parameters X, y and $k = 5$. Append the result to an array labeled `cv_errors`.
  - Call `validation` with parameters X and y. Append the result to an array labeled `val_errors`.

- (20 pts) On a single figure, plot:
    - A line plot: Integers from 1 to 5 (inclusive) for the x-axis, `cv_errors` for the y-axis.
    - A line plot: Integers from 1 to 5 (inclusive) for the x-axis, `val_errors` for the y-axis.
    - Don't forget to put a title, axis labels and a legend. Sample text for these can be seen below.

Since we're randomly shuffling the data, the output will be different each time you run the code. Here are 3 example output plots:




Cross-validation vs Validation
1e7
Cross-validation
Validation
Mean squared error
Shuffle number
1.4
1.2
1.0
0.8
0.6
1
2
3
4
5


Cross-validation vs Validation
1e7
Cross-validation
Validation
Mean squared error
Shuffle number
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
1
2
3
4
5

Now, for the insight section:

The aim of this lab was to showcase the difference between different validation methods. “Cross-validation” is what we worked on the previous lab, and “validation” should be clear to all: It is the simple process of dividing the data into train and test subsets, train the model using one subset, and test for accuracy using the other. Although validation is a simple enough technique to use, the interpretability of the results is not too high: As the graphs indicate, the results vary drastically. As you can imagine, the fluctuation in these results is directly linked to which part of the data is set as test data. On the contrary, cross-validation errors are stable throughout every graph (except for a single point on the second graph, which can be attributed to the randomness of the shuffle). Therefore, when comparing two different regression methods, comparing cross-validation errors will give you a *much* better idea. Here, we only implemented a single regression algorithm (linear regression), but as we advance further in the semester, we will discuss different regression methods. And when we want to compare these to one another, we must use proper validation methods.

We will move on to different multi-dimensional regression techniques in the future labs.