LAB 3 - Multiple Linear Regression

In this lab, we will perform multiple linear regression in Python. To do this, we will need the .csv provided with the assignment on Blackboard.

Again, this is the same .csv file we used in the previous labs. This time, we will investigate the effects of player age, height, mental strength and skill (x_1, x_2, x_3 and x_4) to the player's salary (y).

Instructions:

1. (20 pts) In order to implement multiple linear regression, we need to have our independent variables as a matrix (X). Therefore, you will need to extract each independent variable column ("Age", "Height", "Mental and "Skill" in this case) and concatenate them together (along with a ones column, so you should have 5 columns in total) to form the correct matrix, which should look like this:

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & \dots & x_{k1} \\ 1 & x_{12} & x_{22} & \dots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \dots & x_{kn} \end{bmatrix}$$

Here, instead of having one single variable as a vector, we have multiple vectors forming a matrix. Concatenating vectors can be done in many ways, which we experimented with, while working on the Python exercises. The easiest method would be to use the `column_stack` function within the `numpy` package.

2. (20 pts) For each column in the matrix, we obtain a different coefficient. The aim is to find coefficients $\hat{\beta} = \{\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_n\}$ such that:

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots\dots + \hat{\beta}_n x_n, \text{ where}$$

$x_i,\ i = 1, 2, 3, \dots, n$ is our set of input variables, and
$\hat{\beta}_j,\ j = 0, 1, 2, \dots, n$ is the set of estimated coefficients.

Use the formula below to compute $\hat{\beta}$:

$$\widehat{\boldsymbol{\beta}} = [\boldsymbol{X}'\boldsymbol{X}]^{-1}\boldsymbol{X}'\boldsymbol{y}$$

Note that this is a linear algebra equation and you should use the linear algebra functions within `numpy` to compute the result. Do this in a function which accepts X and y as parameters and returns the vector of coefficients $\hat{\beta}$.

3. (10 pts) Implement a **function** which calculates and returns the MSE (mean squared error) for a given set of predictions. This function should take two parameters: One for the actual y values, the other for the estimated $\hat{y}$ values. It should finally return the MSE as a single floating-point number. The calculation can be found below:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

4. (10 pts) Split the data 80 to 20 (first 80 train, last 20 test). Then:

- Find regression coefficients using train data.
- Find the estimated salary values using $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$, with test data.
- Find and print the MSE for these predictions.

5. (10 pts) Repeat step 4, without splitting the dataset (all 100 rows will act as both train and test data).

6. (10 pts) Change your input matrix X by adding an extra column to it. This extra column should have random integers from -1000 to 1000. You can use the function

```
numpy.random.randint(low, high, size)
```

to create the column.

7. (10 pts) Repeat steps 4 and 5 with the new X and y.

Here is an example of a desired output:

```
------------------- ORIGINAL DATA -------------------

Test error w/ 80-20 split:
MSE: 10733053.490140421

Training error:
MSE: 15733111.677739916

-------------- DATA W/ RANDOM COLN ------------------

Test error w/ 80-20 split:
MSE: 11401785.374856798

Training error:
MSE: 15724187.407532854
```

The first two scores are absolute. The rest, since they include random numbers, is going to be different each time you run the program.

<u>Important note:</u> These instructions require you to implement the calculations manually. Any submissions which bypass such calculations will receive 0 points from corresponding parts.