LAB 2 - Simple Linear Regression (continued)

In this lab, we will repeat our efforts to implement single linear regression on player experience to salary (which was already done in LAB 1), with small extra steps. This time, we will need to split the .csv file into two. First, we will have to *train* the algorithm using one half, i.e. we will use our simple linear regression algorithm on that half. Next, we will see how our estimation fares against the other half, we will *test* our regression line using that one. The .csv file is attached to this assignment in Blackboard.

Instructions:

- (10 pts) Extract the “Experience” and “Salary” columns from the .csv file just like you did in the first lab session. Then, separate the first half (first 20 rows) of both columns, label them `exp_list_1` and `sal_list_1`. For the second half, label them, `exp_list_2` and `sal_list_2`.

- (20 pts) Perform the linear regression algorithm twice (using the `simlin_coef` function you implemented in the previous lab, it would be useful if you imported your previous implementation here):

    - First, give `exp_list_1` and `sal_list_1` as parameters, and label the corresponding coefficients as `b0_1` and `b1_1`.
    - Then, give `exp_list_2` and `sal_list_2` as parameters, and label the corresponding coefficients as `b0_2` and `b1_2`.

- (35 pts) Plot the regression lines. To do this, again, import the `simlin_plot` function you have implemented in the previous lab. Before using it, implement 3 changes to it:
    - It shouldn’t call `show()`,
    - It should create a new figure (window) each time it is called
    - It should return the regression line.

    After implementing these changes;

    - Call `simlin_plot` for parameters `exp_list_1, sal_list_1, b0_2` and `b1_2`. Assign the regression line to a variable labeled as `sal_pred_1`.

- Call `simlin_plot` for parameters `exp_list_2, sal_list_2, b0_1` and `b1_1`. Assign the regression line to a variable labeled as `sal_pred_2`.

- Only call `show()` at the VERY end of your code, so we can see both the plots and the console output without needing to close the plot windows.

- (35 pts) As a final extra step, implement a new function which calculates and displays the R^2 scores. This function should take two parameters: One for the actual y values, the other for the estimated y values. The calculation can be found here:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{TSS} = \sum(y_i - \bar{y})^2$$

Here:

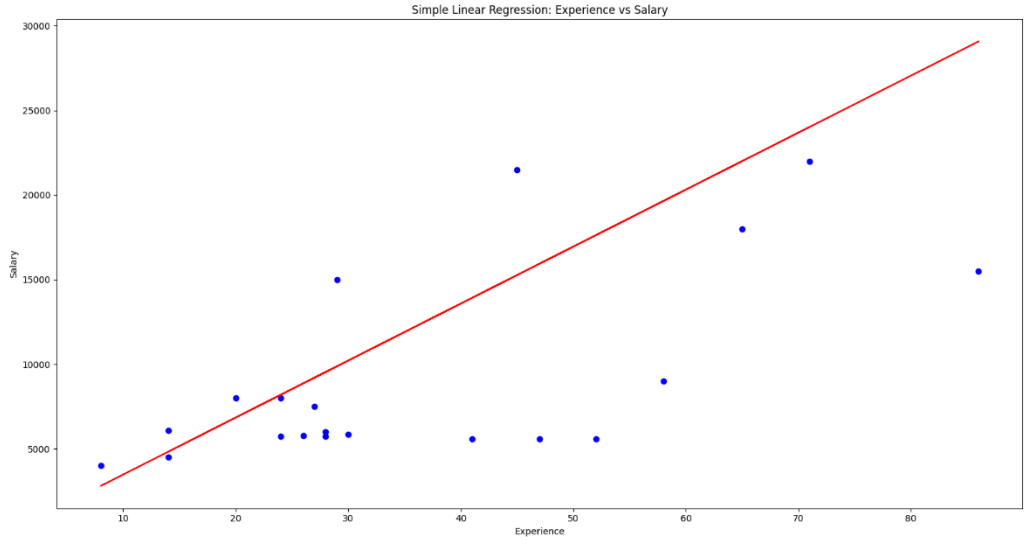
- y_i refers to the i^{th} salary value,
- $\hat{y}_i$ refers to the i^{th} prediction value
- $\bar{y}$ refers to the *average* of all actual salary values.

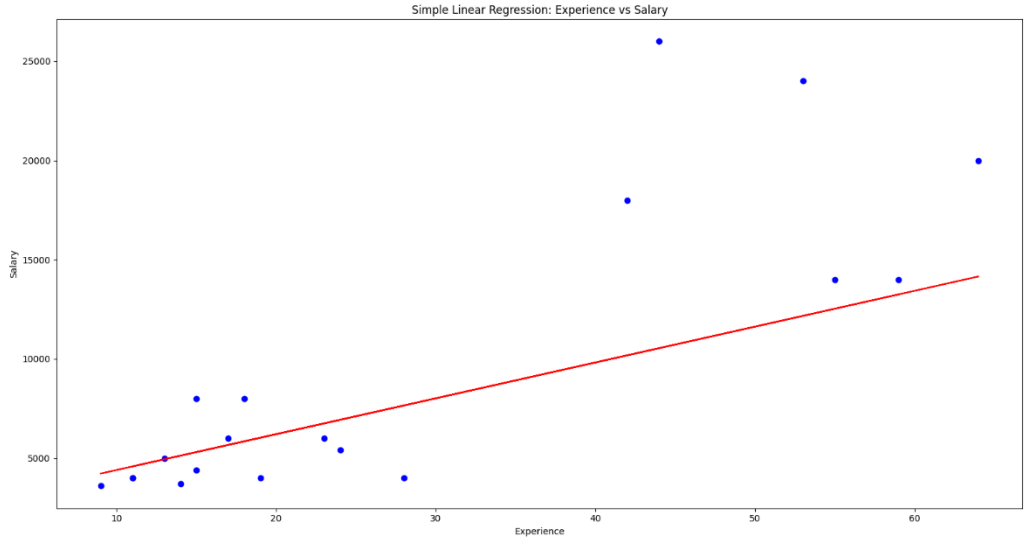
You will call this function twice.
One call will be done by using `sal_list_1` and `sal_pred_1` as parameters.
The other call will be done by using `sal_list_2` and `sal_pred_2` as parameters.

- IMPORTANT NOTE: Again, just like the previous lab, entirely bypassing the regression calculations is not permitted (you can still use `numpy` functions).

Below are output samples for the plots and for R^2 scores:

```
R^2 score: -0.23422810970326546
R^2 score: 0.48005248681601764
```


Simple Linear Regression: Experience vs Salary
Salary
Experience
30000
25000
20000
15000
10000
5000
10
20
30
40
50
60
70
80


Simple Linear Regression: Experience vs Salary
Salary
Experience
25000
20000
15000
10000
5000
10
20
30
40
50
60

Continuing with the insight on this lab session:

As mentioned, in the previous lab, we built a model, but we never tested that model. In standard machine learning tasks, a model is usually tested on a different data, to see how accurate our model is. In this lab, we can see the distinction.

When we create a model using a data, we *train* the data. The data we use is called the *train* set. Then, we compare our model to a different dataset, which is called the *test* set.

We performed two different linear regressions here. One half of the data acted as test data in one regression and acted as train data in the other regression. Same goes for the remaining half.

Showing the results in the form of a numerical value is also very important. Here, we were able to show the results by directly plotting the regression line and the data itself, but in many cases, the data (or the model) will be in a multiple-dimension representation, so plotting the data together with the model will be impossible. In these cases, we need numerical values to see if the model is accurate or not.

We have calculated the R^2 scores for two different cases. This metric compares the estimation (the regression line) with the mean of the data (which is not shown on the graphs, but you can imagine it as a horizontal line at the average value of the y-axis). The score reflects how much of the variance we were able to reduce using our model.

In one of the cases, the R^2 score is negative. This means that the mean of the data has lower variance than our model, which means that our model increased the variance.