LAB 4 - Multiple Linear Regression Using k-fold Cross-Validation

In this lab, we will perform multiple linear regression with using the k-fold cross-validation technique. To do this, we will need the .csv provided in the LAB 4 folder.

The .csv file is the combination of the two different .csv files in the LAB 2 folder. We will perform the same regression as LAB 3 (i.e. we will investigate the effects of age, experience and power to the player's salary), but this time we will perform it multiple times, once for each "fold". A fold is simply a slice of the data, formed by taking several rows from the original matrix.

We will select $k = 10$. This means that we must split our matrix into 10 folds. Since our data has 40 rows in total, a fold consists of 4 rows of a matrix. Follow the instructions below:

(50 pts) You should perform a loop (pay close attention to how the loop iterates!!!), where:

- The next fold (the next 4 rows) is selected as test data, and the rest becomes training data. You can use `numpy.delete()` to form the training data.

- The regression coefficients are calculated using the training data.

- The estimations for the *test* data are found. Simply compute $\hat{y} = X\hat{B}$ where X is the test data. The estimations should be stored inside an array for future use. Label these estimations as "`cv_hat`".

The loop iterates until all folds are used as test data. After the loop finishes, calculate and display MSE for the predictions stored in "`cv_hat`". The calculation for MSE is as follows:

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

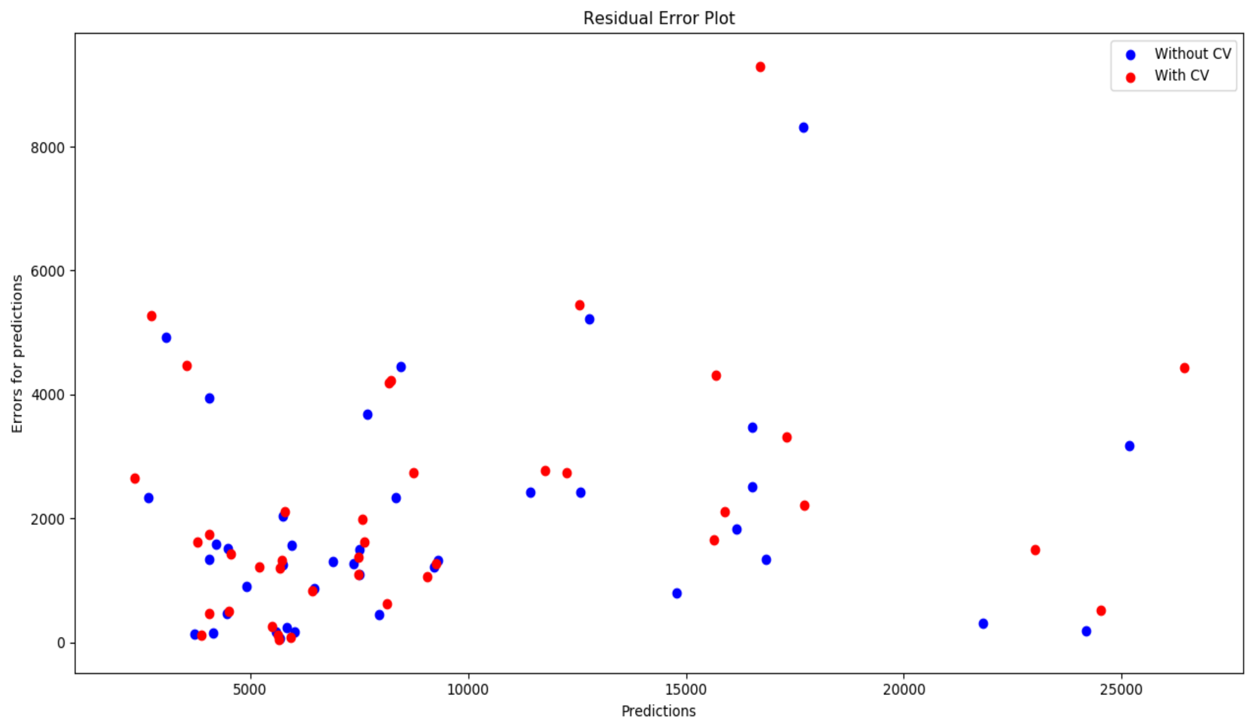
Where $\hat{y}_i$ are the predictions and y_i are the original y (salary) values, and n is the total number of elements.

(15 pts) You should then do the regression one more time, using *all* of the data both as test and train data. Label your predictions as "`y_hat`". Calculate and display the MSE for this set of predictions. The values should be as the following:

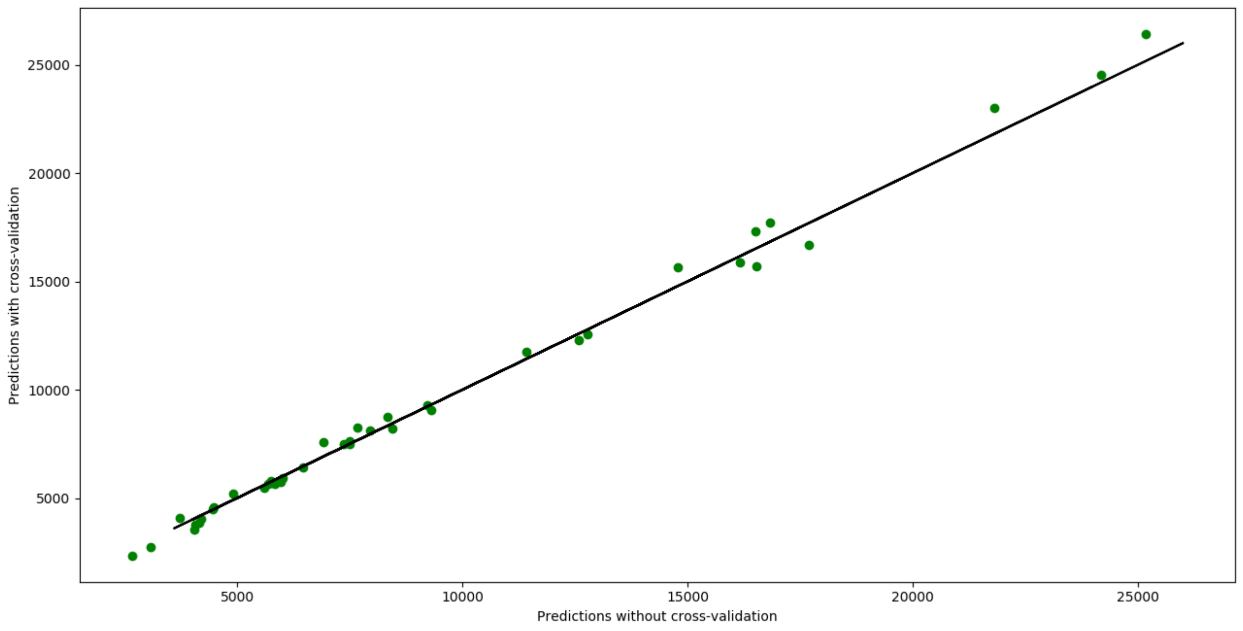
```
MSE with cross-validation: 8083762.961726533
MSE without cross-validation: 6337368.794770615
```

(35 pts) Remember the estimate-error plot you did in the LAB3? Now, you will perform the same plot twice:

- One plot will be done using the prediction values from the first task, "`cv_hat`". Calculate the errors for these predictions, then make a **scatter** plot using the <u>predictions</u> as your x-axis, and the <u>error</u> as your y-axis.
- The other plot will be done using the prediction values from Task 2, "`y_hat`". Again, calculate the error and plot like you did previously.
- Draw both plots and the line on the same figure (window). The output should look like this:



Then, just to see differences between predictions, we will plot "`y_hat`" against "`cv_hat`". In a new window, make a scatter plot with "`y_hat`" as your x-axis, and “`cv_hat`” as your y-axis. To be able to clearly see the differences, also plot a line on the same window at $y = x$. The output can be seen below:



As always, please use `numpy` arrays for any numerical arrays or matrices. Similar to the previous labs, we will use `matplotlib` for plotting purposes. No outside packages are permitted for use except `pandas`.

Continuing with the insight section:

Cross-validation is a very useful technique which allows us to get better understanding about our model, especially if we have a limited sample size. Overfitting/underfitting issues can be monitored efficiently.

Notice that the MSE value for the second task was lower than the first one. This is because we created our model using some data, and if we use the *same* data to test, our error is going to be minimized (that was the goal when creating the model in the first place). In cross-validation, we take some data points as training data, so the rest will be unfamiliar for our model, and the error margin will be undeniably greater. However, we can more accurately determine if our model is effective or not.

In k-fold cross-validation, which we implemented today, we take equal portions of our data (k portions in total) and use them separately for testing. When a portion is used for testing, the rest becomes training data. Leave-one-out cross-validation (LOOCV) is an extreme case where k is equal to the total number of data points we have. In that case, a fold simply consists of a single row.