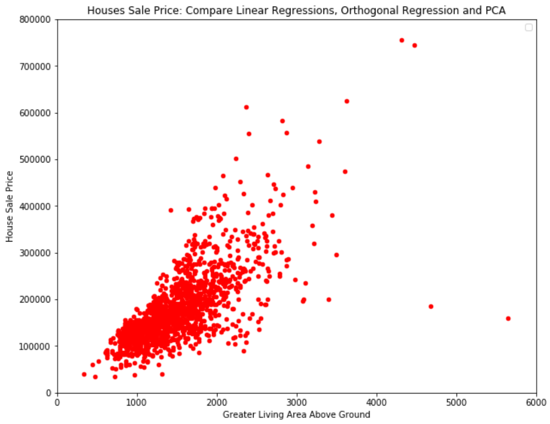
Exercise 2
Principal Components or Regression?

The goal of this exercise is, using a two-dimensional dataset, to compare Principal Component Analysis (PCA) – an unsupervised learning approach - and Linear Regression – a supervised learning approach.
The dataset used can be obtained from a Kaggle competition named “House Prices: Advanced Regression Techniques”. It contains 1460 training data points and 80 features that might help predict the selling price of a house. The dataset is described in details in https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data. A copy of the dataset, called *houseprices.csv*, is also available in the course folder.

For this exercise we will focus on the two following variables of the file *houseprices.csv*:
Saleprice: the house sale price ($)
GrLivArea: above ground living area (square feet)
We will compare the result of applying regression or PCA to this two-variable dataset. We will use the variable name x for *GrLivArea* and y for *Saleprice*. The following is the cross-plot of y vs x associated with the 1460 houses in the dataset.
A Python solution code is associated with the exercise, it is called *PCARegression.py* and is available.



1. Give the mathematical expression of the bias term θ_0 and the slope θ_1 of the regression line for predicting y from x.
   Calculate the regression parameters θ_0 and θ_1.
   Also calculate the regression parameters and the R2 variance score using the *LinearRegression* module in the *sklearn.linear_model* library . Check that your calculations of θ_0 and θ_1 are right.

   The formula for calculating $\boldsymbol{\theta_0}$ and $\boldsymbol{\theta_1}$was given during this morning’s course:

$$\boldsymbol{\theta_0 = \frac{\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)2}\right)\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)}\right)}{\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)2}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)^{2}}}$$

and

$$\boldsymbol{\theta_1} = \frac{\frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)} - \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)}{\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)2}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)^2}$$

Programming these two formulas in a simple Python routine gives:

θ_0 =18569.0259 and θ_1 = 107.1304

***Using the sklearn LinearRegression* module gives us the same values for both parameters and a value for the R2 variance score equal to 0.50, which indicates a correlation coefficient of $\rho = \sqrt{0.50} = 0.71$.**

2. Give the mathematical expression of the bias term θ_0' and the slope θ_1' of the regression line for predicting x from y.
   Calculate θ_0' and θ_1' .
   Also calculate the regression parameters and the R2 variance score using the *LinearRegression* module in the *sklearn.linear_model* library . Check that your calculations of θ_0' and θ_1' are right and plot the line.

   The formula to calculate θ_0' and θ_1' is similar to the one of the previous question, except that we exchange the variables x and y:

$$\boldsymbol{\theta_0'} = \frac{\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)2}\right)\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)}\right)}{\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)2}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)^2}$$

   and

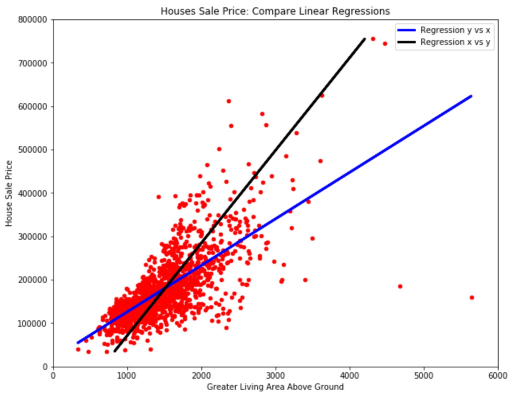
$$\boldsymbol{\theta_1'} = \frac{\frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)} - \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)}{\left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)2}\right) - \left(\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\right)^2}$$

   Which gives us, using the same Python routine as that used for question 2:

   θ_0' = 667.4377 and $\theta_1' = 0.0047$

   Using the *sklearn LinearRegression* module gives us the same values for both parameters and a figure for the R2 variance score of 0.50, as for the regression above. This is expected, since we saw above that the R2 variance score is equal to the square of the correlation coefficient, which is the same for x vs y or y vs x!

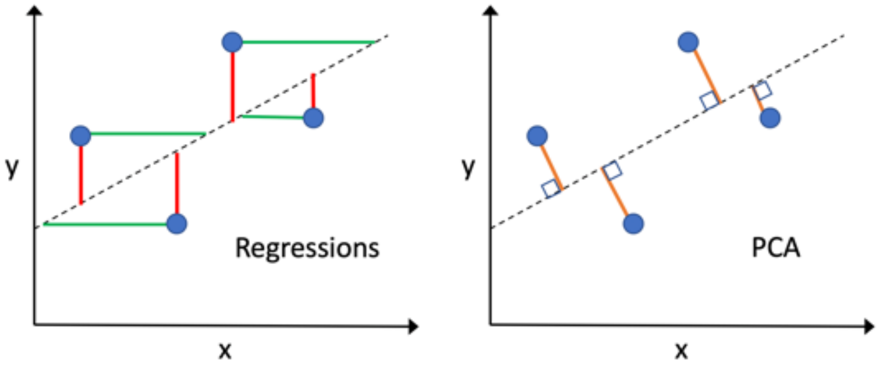
3. Plot the two regression lines calculated in the previous questions, and cross-plot the dataset on the same figure.



We observe that the two lines intersect at the mean of both variables.
An interesting property is that the product of the slopes of the two lines is equal to the square of the correlation coefficient, which is also the R2 variance score:

$$\mathbf{107.1304 \times 0.0047 = 0.50}$$

4. With regression, as seen in the two previous questions, what is minimized is the sum of squares of the differences between the data values and the regression lines. This can be summarized in the picture below (the sum of the squares of the red lines are minimized by regression of y vs x, the sum of the squares of the green lines are minimized by regression of x vs y). On the other hand, PCA minimizes the sum of the squares of the orange lines, that is the perpendiculars to the calculated principal axis (we can also say, as discussed at the end of the morning, that PCA maximizes the variance of the data projected on the principal axis).



There are two ways to calculate the slope of the PCA line in two dimensions (it is enough to calculate the slope as the second parameter can be derived from the fact that, as was the case for the two regression lines, the PCA line goes through the point associated with the mean of the two variables).
The first approach is to calculate it analytically using the method known as Orthogonal Regression, where the sum of the squares of the orange segments in the above figure is minimized. It can be demonstrated that the formula for the slope is:

$$\theta_1'' = \frac{(\sum_{i=1}^{m} V_i^2 - \sum_{i=1}^{m} U_i^2) + \sqrt{(\sum_{i=1}^{m} V_i^2 - \sum_{i=1}^{m} U_i^2)^2 + 4(\sum_{i=1}^{m}(U_i V_i)^2)}}{2\sum_{i=1}^{m}(U_i V_i)}$$

Where U and V are the centered variables x and y, that is the variables x and y after substraction of their mean:

$$U = x - m_x \text{ and } V = y - m_y$$

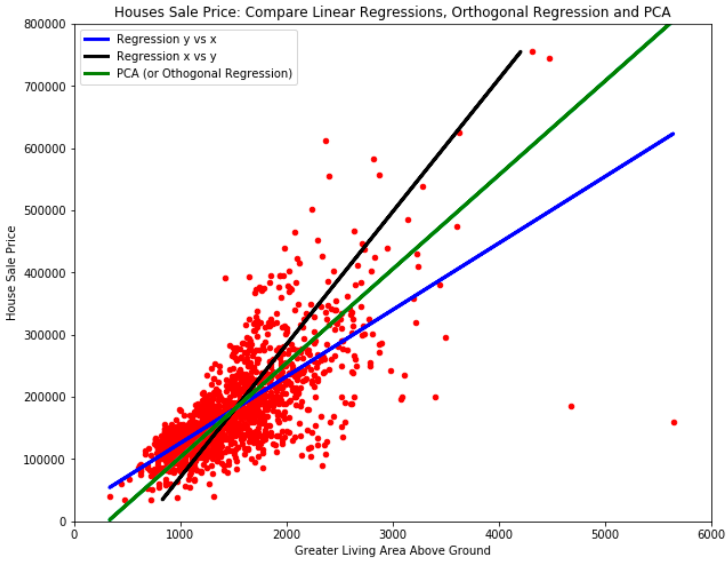
The second approach is to calculate it using (for instance) the *PCA* module in the *sklearn.decomposition* library. With this second approach, it is important to first standardize *Saleprice* and *GrLivArea*. Use the two approaches and verify that you obtain the same result.

Using either the equation given above for $\boldsymbol{\theta_1''}$ or the *sklearn PCA* module, we obtain the same result:

$$\boldsymbol{\theta_1'' = 1}$$

(indeed it is easy to check analytically that any vector of slope 1 is an eigenvector of the correlation matrix, with the associated eigen value equal to 1+the correlation coefficient). Note that the "Explained Variance" calculated by the sklearn PCA module is not to be confused with the R2 variance calculated above.

5. Compare the three lines on the same cross-plot. What do you observe?



We observe that the PCA line lies just between the two previously calculated regression lines. This is expected as, in the PCA formalism, $\boldsymbol{x}$ and $\boldsymbol{y}$ play a symmetric role. As mentioned before, PCA (or orthogonal regression) is an unsupervised technique: the algorithm is presented with a set of points in the plane, and it has to project this set on points on the line that loses a minimum

amount of information after the points have been projected on the line. On the other hand, with regression, the algorithm is given either a set of x values labelled by y (this is regression of y vs x, where we have to predict the best value of the label y given x) or a set of y values labelled by x (this is regression of x vs y, where we predict the best value of the label x given y).

The code *PCARegression.py* provides a solution to the questions above