LAB 7 - Polynomial Regression

In this lab, we are going to implement polynomial regression. This technique, as we shall see, is very similar to standard multiple linear regression.

Our input file is now different: It is a dataset related to bluegill fish. It contains only two columns: The age and the length of a bluegill fish. Our aim is to model the relationship between the two. Age is going to be the input (x), while length is going to be the output (y).

- (50 pts) Implement the cubic spline regression in a function with three parameters: x, y and the *degree* of the polynomial which we're modeling the relationship as. Inside this function:

  - (40 pts) Create your X matrix:

    Just like multiple linear regression, we need an input matrix. But this time, aside from our x, our other independent variables are going to be some function of x. See below for details:

$$\left.\begin{aligned} x_1 &= x \\ x_2 &= x^2 \\ x_3 &= x^3 \\ &\vdots \\ x_d &= x^d \end{aligned}\right\} \Longrightarrow \mathrm{X} = \begin{array}{ccccc} & x & x^2 & & x^d \\ & \downarrow & \downarrow & & \downarrow \\ \end{array}$$

$$\begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{d1} \\ 1 & x_{12} & x_{22} & \cdots & x_{d2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{dn} \end{bmatrix}$$

where d is the degree of the polynomial, provided as a parameter for the function.

- (10 pts) Calculate coefficients, and return the regression line:

  You can calculate the coefficients using:

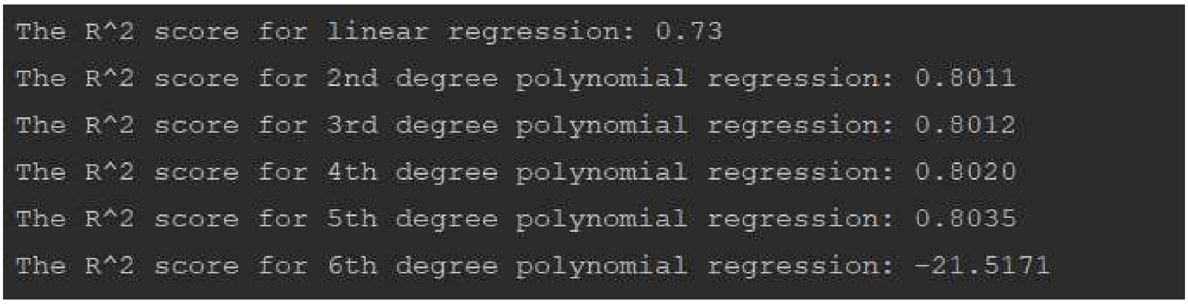
  $$\widehat{\boldsymbol{B}} = \ [\boldsymbol{X'X}]^{-\boldsymbol{1}}\boldsymbol{X'y}$$

  After calculating the coefficients, calculate the regression line as your prediction and return it. You can calculate the line using:

  $$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{B}}$$

- (10 pts) In the main part of the script, read your data sheet, extract your x and y vectors. Call the function you implemented six times, using the same x and y, but different degree values as 1, 2, 3, 4, 5 and 6. Store each set of predictions in a different array.

- (10 pts) Calculate and display the R^2 score for all 6 set of predictions. The results should look like the following:



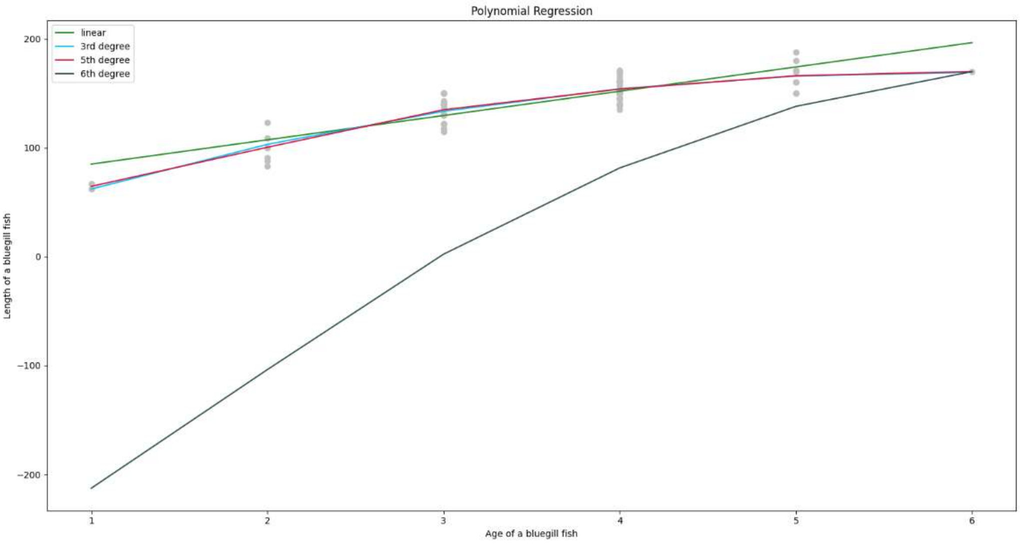
```
The R^2 score for linear regression: 0.73
The R^2 score for 2nd degree polynomial regression: 0.8011
The R^2 score for 3rd degree polynomial regression: 0.8012
The R^2 score for 4th degree polynomial regression: 0.8020
The R^2 score for 5th degree polynomial regression: 0.8035
The R^2 score for 6th degree polynomial regression: -21.5171
```

- (30 pts) Plotting:
  - Before plotting, we must sort predictions according to x (fish age).
    To sort the data, do the following:
    - Call `numpy.argsort(`x`)` and store the result. The result is a set of indices, giving you the sorted locations of x.
    - Combine x, y and every set of predictions together in one matrix.
    - Change your matrix to `matrix[set_of_indices_from_step_1]`.
  - Plot the data points (x and y, the values you initially extracted from the .csv file) as a scatter plot.

- Plot all regression lines **except the 2nd and 3rd order polynomial regression lines,** all in different colors, on the same window. In all line plots, the x-axis is x, and the y-axis corresponds to the predictions you calculated.

- Put a corresponding title, axis labels and legend.

  You should get a result like the following:



Continuing with the insight section:

In this lab, we implemented polynomial regression. As you experienced, this method uses the same computations as linear regression (called the *least squares* method), except our input variables are duplicates of each other, either squared, or cubed etc., depending on the total degree of the polynomial function. These duplicate variables are also called *dummy variables*.

Since we're using the same method, you can see that when we set the degree as 1, the procedure is identical to linear regression.

A 3rd or 4th degree polynomial is usually the maximum degree of choice for this task, because the larger the degree goes, the more flexible the curve becomes. This would result in very strange shapes. The 6th degree polynomial is a demonstration of this phenomenon, and if you go beyond that, you will encounter even stranger curves.

We can use cross-validation or other techniques to choose approximately optimal degree values. Or, we can simply try every value out separately!