Homework 9 *(Due: Nov 15)*
Python Programming for Data Science - COSC 3360

Department of Computer Science and Electrical Engineering

Fall Semester, 2022

Exercises

*Create a **New Project** for every exercise. Take a screenshot of the source code along with its output and place the **source code** and the **screenshot** in a **zipped folder** named **LastNameFirstName_HW9***

Exercise 1

Given the following **house square feet**: $250, 500, 1000, 2000, 3000, 4000$ and the following **house prices**: $50000, 100000, 200000, 400000, 600000, 800000$. Create **noise** derived from a standard normal (Gaussian) distribution function (i.e., one with 0 mean and 1 standard deviation). Multiply the noise derived by 30000 and then add it to the dependent variable. Use **Ordinary Least Squares** to find the regression line parameters, estimate the **SSE** (Sum of Squared Error), and find the **Correlation Coefficient**. **Plot** the data points with and without noise, along with the regression lines with and without noise, and **predict** house prices for the following square feet: **200, 1250, 2710, 5100**

Note 1: The following line of code produces 6 numbers from a normal distribution with 0 mean and 1 standard deviation: *randomNumbers = np.random.normal(0.0, 1.0, 6)*

Note 2: Declare your arrays as: *np.float64()* instead of *np.array* or use e.g., *np.array([250.])* in at least one element so as to be treated as a *float* array and multiplying it with an *int* array will also result in a *float* array

Note 3: Do not use any built-in functions, apart from *np.mean()*, to estimate the: *regression line, SSE, correlation coefficient*; you can use them, though, for verification with your own results

Exercise 2

Based on Ex. 1, find the **Least Squares Regression Line** using the *Matrix Algebra* method. Verify whether you get the same regression line parameters as in Ex. 1. **Plot** the data points, after noise addition, along with the regression line

Note: You can use the *transpose()*, *np.matmul()*, and *np.linalg.inv()* functions to get the *transpose*, *multiply* matrices, and get the *inverse* of a matrix, respectively

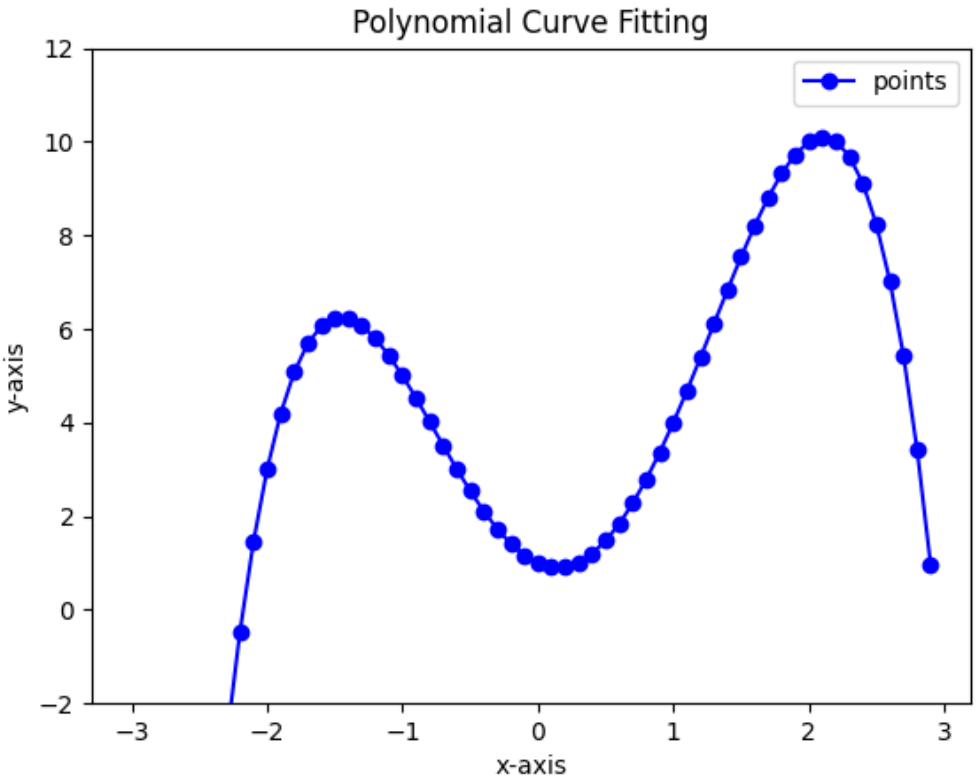
See overleaf

Exercise 3

Perform **Polynomial Curve Fitting** on the following dataset: (-2, 3), (-1, 5), (0, 1), (1, 4), (2, 10). **Print** the coefficients and **plot** the graph of the polynomial function as shown in the Figure below

Note 1: Do not use any built-in functions such as *polyfit()* or *polyval()*. You can use the functions *np.matmul()* and *np.linalg.inv()* to *multiply* matrices and get the *inverse* of a matrix, respectively

Note 2: Your algorithm should be able to work with any dataset not just a dataset of 5 data points



Note: Submit through **Canvas**