Submission Date: **5.11.2025**

1 Bias-Variance Trade-off

You are working as a data scientist for a very important company and, suddenly, you are tasked to train a *polynomial* model on the data from Figure 1.

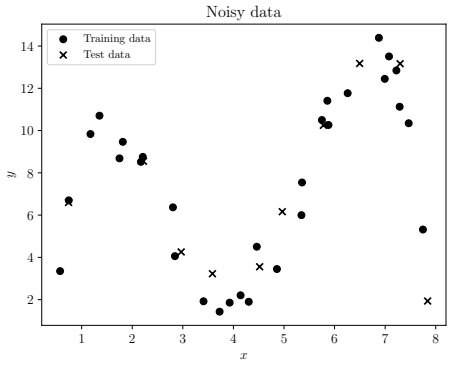

Figure 1: Noisy Data.

1. Only from looking at the plotted data, which degree of a polynomial would be your best guess that would fit the data without over- or underfitting it? Why?

2. Predict how the bias and variance (high / low) and fit (under / over) would be for:

   - A constant model (i.e. a horizontal line)
   - A linear model
   - A polynomial model of degree 10

Using Python and SciKit-Learn: Included in the exercise are two files (`data_train.csv` and `data_test.csv`) containing the noisy data in a csv format, where the first column represents the input features x and the second column contains the measured outputs y. Additionally, there is a jupyter notebook that contains some code snippets to help you get started. The notebook requires a virtual environment using anaconda or venv with numpy, sklearn, jupyter and matplotlib.

3. Use `numpy` to load the two splits of the dataset and use `matplotlib` to try to recreate the plot from Figure 1.

4. Using the `sklearn.linear_model.LinearRegression` class, fit a linear model to the data in the `data_train.csv` file. Compute the MSE error

for the data in `data_test.csv` and plot the predicted line together with the original data.

5. Create a class that can fit a polynomial model of an arbitrary degree. For example `m = PolynomialModel(degree=23)`. Include a `fit(x_train, y_train)` method for fitting some training data and a `predict(x)` method for performing inference for the values of unseen data. Using your class, fit models with *degrees* $\in \{0, 1, 2, 3, 4, 5, 20\}$ and plot the fitted curves along with the original data.
   Hint: You can use `sklearn.linear_model.LinearRegression` and `sklearn.preprocessing.PolynomialFeatures` together to define non-linear models by combining a linear model with basis functions.

6. Create a function `measure_bias_var_mse(model, x_train, y_train, x_test, y_test, n)` that *estimates* the bias, variance and, mean square error of different models.
   Hints: The function should:

   (a) sample *with replacement* n datasets $\left\{\mathcal{D}_1^{(\text{train})}, \ldots, \mathcal{D}_n^{(\text{train})}\right\}$ from the training dataset,

   (b) fit a model for each re-sampled dataset $f_i\left(x;\ \mathcal{D}_i^{(\text{train})}\right)$

   (c) generate predictions over the test data with each model and store the predictions in some array $pred_{i,\,j} = \hat{f}\left(x_i^{(\text{test})};\ \mathcal{D}_j^{(\text{train})}\right)$

   (d) compute the *expected prediction model*:

$$\begin{aligned}\bar{f}(x^{(\text{test})}) &= \underset{\mathcal{D}^{(\text{test})} \sim \mathcal{P}^n}{\mathrm{E}}\left[\hat{f}\left(x^{(\text{test})};\ \mathcal{D}\right)\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} \hat{f}\left(x^{(\text{test})};\ \mathcal{D}_i^{(\text{train})}\right) \end{aligned} \tag{1}$$

   (e) compute the *bias* using the expected prediction model:

$$\begin{aligned}\text{bias} &= \underset{x,y}{\mathrm{E}}\left[\left(\bar{f}\left(x^{(\text{test})}\right) - y^{(\text{test})}\right)^2\right] \\ &= \frac{1}{|\mathcal{D}^{(\text{test})}|}\sum_{i=1}^{|\mathcal{D}^{(\text{test})}|}\left(\bar{f}\left(x_i^{(\text{test})}\right) - y_i^{(\text{test})}\right)^2 \end{aligned} \tag{2}$$

   (f) compute the *variance* using the expected prediction model:

$$\begin{aligned}\text{variance} &= \underset{x,\mathcal{D}}{\mathrm{E}}\left[\left(\hat{f}(x;\ \mathcal{D}) - \bar{f}(x)\right)^2\right] \\ &= \frac{1}{n \cdot |\mathcal{D}^{(\text{test})}|}\sum_{i=1}^{|\mathcal{D}^{(\text{test})}|}\sum_{j=1}^{n}\left(\bar{f}\left(x_i^{(\text{test})}\right) - \hat{f}\left(x_i^{(\text{test})};\ \mathcal{D}_j^{(\text{train})}\right)\right)^2 \end{aligned} \tag{3}$$

(g) compute the *average mean square error*:

$$\text{MSE} = \frac{1}{n \cdot |\mathcal{D}^{\text{test}}|} \sum_{i=1}^{|\mathcal{D}^{(\text{test})}|} \sum_{j=1}^{n} \left(\hat{f}\left(x_i^{(\text{test})}; \mathcal{D}_j^{(\text{train})} \right) - y_i^{(\text{test})} \right) \tag{4}$$

7. Using your function for estimating the MSE, bias and variance, compute them for models with *degrees* $\in \{0, 1, \ldots, 10\}$. Generate a plot of the MSE, bias and variance, where the x axis represents the degrees of the polynomial model, and the y axis are the estimated metrics. Do the computed values and plot match your predictions from question 2.?