Analysis of Regression and Resampling Methods

Håkon Olav Torvik, Vetle Vikenes and Sigurd Sørlie Rustad

University of Oslo
Norway
October 6, 2021

CONTENTS

INTRODUCTION

Regression analysis is a statistical method for fitting a function to data. It is useful for building mathematical models to explain observations. There are several regression methods to achieve this, all with their strengths and weaknesses. We will in this paper study three different methods; ordinary least squares, Ridge and Lasso regression. All the code, results and instructions on running the code can be found in our GitHub repository here[1]

Larger datasets contain more information, giving more accurate models. However, real-world datasets usually have a fixed size, and getting more is practically impossible. For smaller datasets it is then useful to have tools mitigating the effects of little data. Resampling methods does this by running the same data through the regression, without over-fitting the model to the sample data. In addition to the regression methods, we will also study the effect of bootstrapping and cross-validating the data.

In order to study this, we need data to analyze. We will use two datasets to study the different methods. One dataset we will generate ourselves using the Franke function, given by equation (1). To make the data more realistic we also add normally distributed noise. The other data is real terrain data from here[2]. Our model for both datasets will be a polynomial fit with an x and y dependence of the form $[x, y, x^2, y^2, xy, \dots]$.

$$\begin{aligned} f(x,y) =& \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}\right)+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49}-\frac{(9x+1)^2}{49}-\frac{(9y+1)}{10}\right) \\ &+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4}-\frac{(9y-3)}{4}\right)-\frac{1}{5}\exp\left(-(9x-4)^2-(9y-7)^2\right) \end{aligned} \tag{1}$$

EXERCISE 1: ORDINARY LEAST SQUARE (OLS) ON THE FRANKE FUNCTION

Using the Franke Function (equation (1)) we generate our dataset. We pick $n = 100$ randomly distributed points for x and y, where our domain is $x, y \in [0, 1]$. As mentioned in the introduction we add stochastic noise to the function, using the normal distribution $\epsilon \cdot N(0, 1)$. Here ϵ is just a variable we can use to scale the noise. Choosing $\epsilon = 0.2$, gives us the dataset visualized in figure 1. We can choose to scale our data, however, this will have no effect when performing ordinary least squares (OLS). We do decide to scale our data by subtracting by the mean value (standard scaling), however this is only done because it makes our program more universal when performing other regression methods.

When we do OLS regression we assume that our data follows a model such that

$$\mathbf{z} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2}$$

Where $\mathbf{z}$ is our data, X is our design matrix given by our model, $\boldsymbol{\beta}$ are parameters in the model and $\boldsymbol{\epsilon}$ is stochastic noise. One challenge in our case is that our data is two-dimensional, i.e. $\mathbf{z}$ has dimensions (n, n). We create a workaround by mapping our two-dimensional data, to a one

[1] https://github.com/sigurdru/FYS-STK4155/tree/main/project1

[2] https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2021/Project1/DataFiles

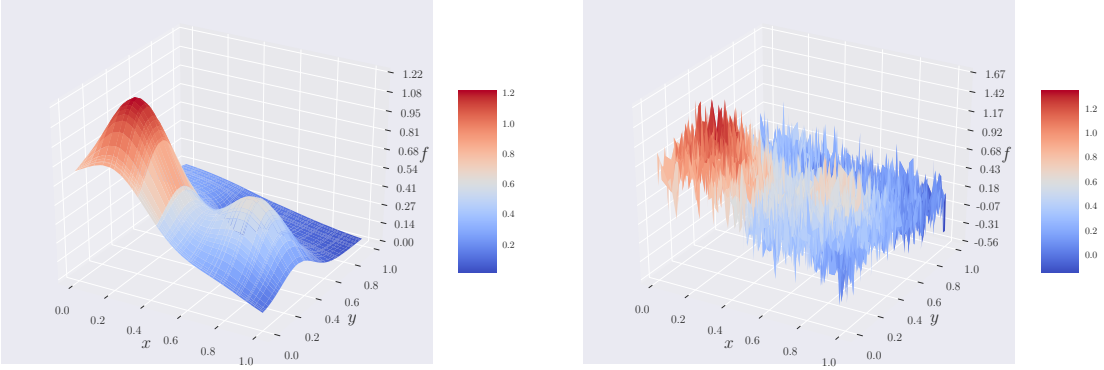

Figure 1. Here we have displayed our data along with the analytical function. The left figure shows the analytical Franke function plotted, and the right hand figure is the Franke function plotted with noise, i.e. the data we use.

dimensional $(n^2, 1)$ vector. This is an important note, and we will do this throughout the report. Note that our design matrix X will have dimensions (n^2, p) where p are the number of features.

As mentioned in the introduction our model will be a polynomial fit with an x and y dependence of the form $[x, y, x^2, y^2, xy, \dots]$. This gives us $p = (P+1)(P+2)/2$ number of features, where P is our maximum polynomial degree. Now for a given model

$$\tilde{\boldsymbol{z}} = X\tilde{\boldsymbol{\beta}},$$

we will have a mean square error (MSE) given by

$$C(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=0}^{N-1}(z_i - \tilde{z}_i)^2 = \frac{1}{N}\left\{(\mathbf{z} - X\tilde{\boldsymbol{\beta}})^T(\mathbf{z} - X\tilde{\boldsymbol{\beta}})\right\}.$$

We call $C(\boldsymbol{\beta})$ our cost function, and is the quantity we want to minimize. The minima can be found by doing the derivative with respect to all the parameters, and set the result to zero

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 = X^T(\boldsymbol{z} - X\tilde{\boldsymbol{\beta}}).$$

We can rewrite this as

$$X^T\mathbf{z} = X^TX\tilde{\boldsymbol{\beta}} \implies \tilde{\boldsymbol{\beta}} = (X^TX)^{-1}X^T\mathbf{z} \equiv \hat{\boldsymbol{\beta}}$$

Thus $\hat{\boldsymbol{\beta}}$ are the parameters that minimizes the cost function. Now that we have an expression for the optimal parameters, we also want to evaluate how good our results are. We do this by splitting our data into two sets, one set for training (80%) and one set for testing (20%). We can then find the optimal parameters ($\hat{\boldsymbol{\beta}}$) using the train data, and then compare our test data to the actual data.

There are many quantities we can use to evaluate our results, however we have decided to calculate the MSE and R^2 score. MSE is given by equation (3) and the R^2 score by equation (4).

$$MSE(\mathbf{z}, \tilde{\mathbf{z}}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 \tag{3}$$

$MSE(\mathbf{z}, \tilde{\mathbf{z}})$ is the mean square between the datapoints $\mathbf{z}$ and $\tilde{\mathbf{z}}$.

$$R^2(\mathbf{z}, \tilde{\mathbf{z}}) = 1 - \frac{\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1}(z_i - \bar{z}_i)^2} \tag{4}$$

Here $R^2(\mathbf{z}, \tilde{\mathbf{z}})$ is the R^2 score between the datapoints $\mathbf{z}$ and $\tilde{\mathbf{z}}$. With this we can plot the quantities as a function of polynomial degree. We expect the MSE to always decrease and R^2 to increase for the train data. We expect the same for the test data, however at a certain number of polynomial degree we might se overfitting. This would make our error increase and R^2 score decrease. The results are plotted in figure 2.

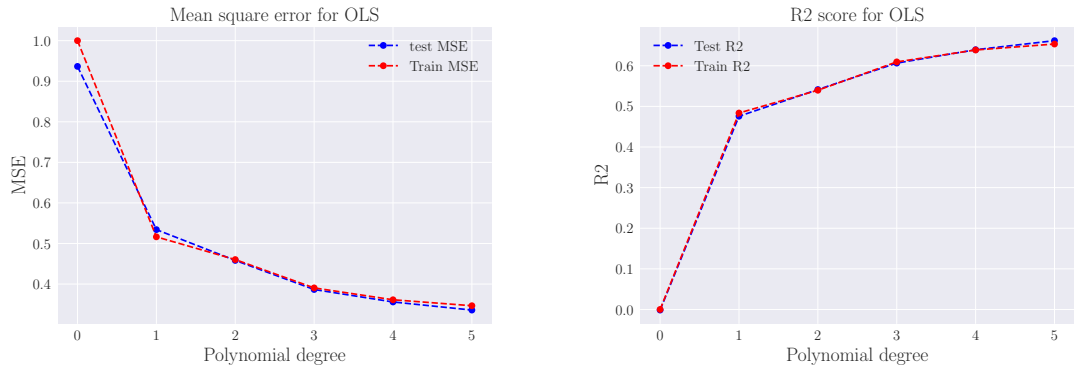

Figure 2. Here we have plotted the MSE and R^2 score for OLS. The left figure shows the MSE and right shows the R^2 score. The red dotted line is from the train data, and the blue dotted line is from the test data.

From figure 2 we see that the MSE strictly decrease and R^2 score increase. This indicates that we have not reached a point of overfitting yet.

It is also interesting to look at the variance in the parameters. We know that for a set of optimal parameters

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{z},$$

We have the expectation value

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \mathbb{E}[(X^T X)^{-1} X^T \mathbf{z}] = (X^T X)^{-1} X^T \mathbb{E}[\mathbf{z}] = (X^T X)^{-1} X^T X \hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}.$$

With this we can calculate the variance in $\hat{\boldsymbol{\beta}}$.

$$\begin{aligned}
\text{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left[(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])^T\right] \\
&= \mathbb{E}\left[((X^T X)^{-1} X^T \mathbf{z} - \hat{\boldsymbol{\beta}})((X^T X)^{-1} X^T \mathbf{z} - \hat{\boldsymbol{\beta}})^T\right] \\
&= (X^T X)^{-1} X^T \mathbb{E}\left[\mathbf{z}\mathbf{z}^T\right] X (X^T X)^{-1} - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= (X^T X)^{-1} X^T \left[X\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T X^T + \sigma^2\right] X (X^T X)^{-1} - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T + \sigma^2 (X^T X)^{-1} - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T = \sigma^2 (X^T X)^{-1}
\end{aligned}$$

Here we used $\mathbb{E}(\mathbf{z}\mathbf{z}^T) = X\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T X^T + \sigma^2\mathbb{1}$, where $\sigma^2 = \epsilon = 0.2$ is the variance of the noise and $\mathbb{1}$ is the identity. Now we have an estimate for the variance in parameter $\hat{\boldsymbol{\beta}}_j$ given by

$$\boldsymbol{\sigma}^2(\hat{\boldsymbol{\beta}}_j) = \sigma^2[(X^T X)^{-1}]_{jj} \tag{5}$$

Using equation (5) we can plot the confidence interval for the different parameters, for different polynomial degrees.

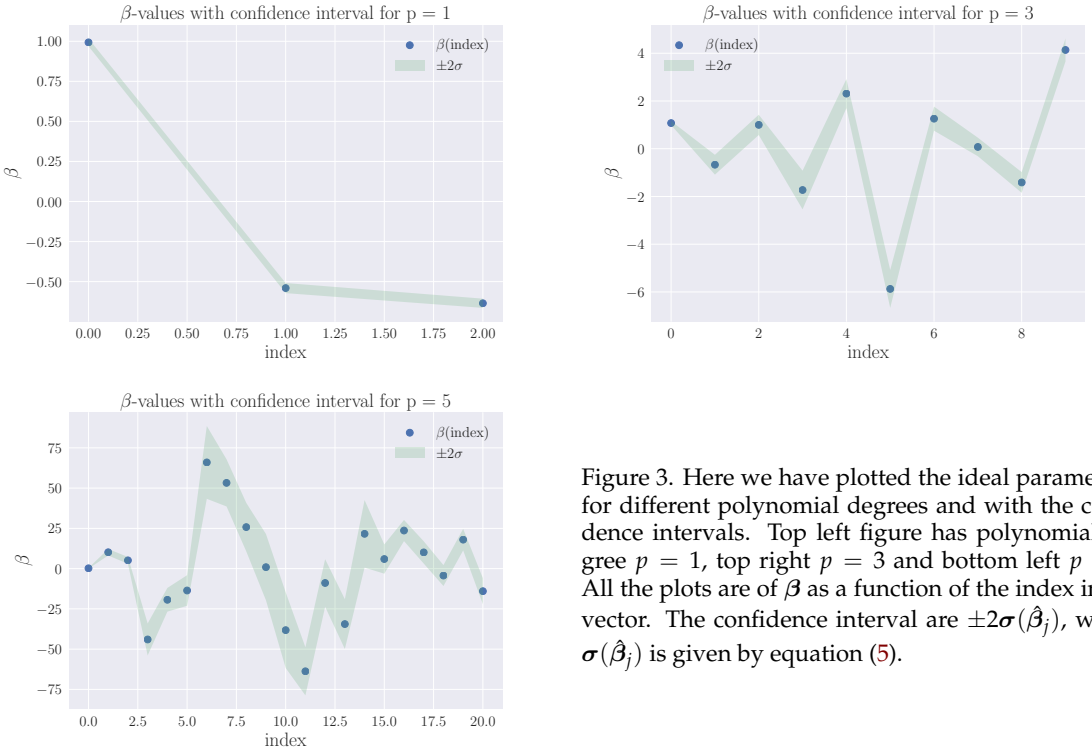

Figure 3. Here we have plotted the ideal parameters, for different polynomial degrees and with the confidence intervals. Top left figure has polynomial degree $p = 1$, top right $p = 3$ and bottom left $p = 5$. All the plots are of $\boldsymbol{\beta}$ as a function of the index in the vector. The confidence interval are $\pm 2\boldsymbol{\sigma}(\hat{\boldsymbol{\beta}}_j)$, where $\boldsymbol{\sigma}(\hat{\boldsymbol{\beta}}_j)$ is given by equation (5).

EXERCISE 2: BIAS-VARIANCE TRADE-OFF AND RESAMPLING TECHNIQUES

With our data model given in equation (2) we find our model by considering the cost function, given as

$$C(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=0}^{N-1}(z_i - \tilde{z}_i)^2 = \mathbb{E}[(\mathbf{z} - \tilde{\mathbf{z}})^2]$$

where $\mathbb{E}$ is the expected value.

We can show that this can be written as [3]

[3] The derivation is given in Appendix A

$$\mathbb{E}[(\mathbf{z}-\tilde{\mathbf{z}})^2] = \frac{1}{N}\sum_i (f_i - \mathbb{E}[\tilde{\mathbf{z}}])^2 + \frac{1}{N}\sum_i (\tilde{z}_i - \mathbb{E}[\tilde{\mathbf{z}}])^2 + \sigma^2 \tag{6}$$

where f_i is the true data value at point i.

In equation (6) the first term represents the square of the bias, the second term represents the variance while the last term represents the variance of the irreducable error ϵ. When performing linear regression the variance is a measurement on how much our model changes with different training sets. High variance will therefore occur if a different training set resulted in very different values of the individual estimators, $\boldsymbol{\beta}$. This will be the case for overfitting when our model is essentially trying to reproduce variations from the noise. The bias provides information about the difference between our model and the true data values. If our model is missing out on underlying structures in our data we would get a high bias. High bias will thus be the case for an underfitted model. Our goal is therefore to minimize the bias and variance in our model.

EXERCISE 3: CROSS-VALIDATION AS RESAMPLING TECHNIQUES, ADDING MORE COMPLEXITY

EXERCISE 4: RIDGE REGRESSION ON THE FRANKE FUNCTION WITH RESAMPLING

EXERCISE 5: LASSO REGRESSION ON THE FRANKE FUNCTION WITH RESAMPLING

EXERCISE 6: ANALYSIS OF REAL DATA

Appendix A: Bias-variance Decomposition

We assume that our true data is generated from a noisy model with nromally distributed noise ϵ with a mean of zero and standard deviation σ^2, i.e.

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

We have approximated this function with our design matrix $\mathbf{X}$ and our parameters $\boldsymbol{\beta}$ such that our model becomes $\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$, where the values of $\boldsymbol{\beta}$ were obtained by optimizing the mean squared error via the cost function, given by

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right]$$

where $\mathbb{E}$ is the expected value.

We want to show that the above expression can be written as

$$\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{\mathbf{y}}])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{\mathbf{y}}])^2 + \sigma^2$$

We begin by inserting our model expression for $\mathbf{y}$ and adding and subtracting $\mathbb{E}[\tilde{\mathbf{y}}]$ inside the expected value, before we square the expression.

$$\begin{aligned}\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] &= \mathbb{E}\left[(f(\mathbf{x})+\boldsymbol{\epsilon}-\tilde{\mathbf{y}}-\mathbb{E}[\tilde{\mathbf{y}}]+\mathbb{E}[\tilde{\mathbf{y}}])^2\right] = \mathbb{E}\left[((f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])+\boldsymbol{\epsilon}+(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}}))^2\right] \\ &= \mathbb{E}\left[(f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])^2+\boldsymbol{\epsilon}^2+(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}})^2\right] \\ &\quad + \mathbb{E}\left[2\boldsymbol{\epsilon}(f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])+2\boldsymbol{\epsilon}(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}})+2(f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}})\right]\end{aligned}$$

where the cross terms have been written on a separate line since the expected value is linear. Next we will focus on the cross-terms. Since $\boldsymbol{\epsilon}$ is normally distributed, it's expected value is simply the mean, which is zero in our case. The two cross terms involving $\boldsymbol{\epsilon}$ is therefore zero, so we only need to consider

$$\mathbb{E}\left[(f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}})\right] = \mathbb{E}\left[f(\mathbf{x})\mathbb{E}[\tilde{\mathbf{y}}]\right]-\mathbb{E}\left[f(\mathbf{x})\tilde{\mathbf{y}}\right]-\mathbb{E}\left[\mathbb{E}[\tilde{\mathbf{y}}]\mathbb{E}[\tilde{\mathbf{y}}]\right]+\mathbb{E}\left[\tilde{\mathbf{y}}\mathbb{E}[\tilde{\mathbf{y}}]\right]$$

Since the expected value of an expected value is just the expected value itself the last two terms in the above equation both become $\mathbb{E}[\tilde{\mathbf{y}}]^2$, canceling each other out. Using that $f(\mathbf{x})$ is a deterministic function, we have $\mathbb{E}[f(\mathbf{x})] = f(\mathbf{x})$. Expressing $f(\mathbf{x})$ in terms of its expected value, we can write the first two terms in the above equation as

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{x})\mathbb{E}[\tilde{\mathbf{y}}]\right]-\mathbb{E}\left[f(\mathbf{x})\tilde{\mathbf{y}}\right] &= \mathbb{E}\left[\mathbb{E}\left[f(\mathbf{x})\right]\mathbb{E}[\tilde{\mathbf{y}}]\right]-\mathbb{E}\left[\mathbb{E}\left[f(\mathbf{x})\right]\tilde{\mathbf{y}}\right] \\ &= \mathbb{E}\left[f(\mathbf{x})\right]\mathbb{E}[\tilde{\mathbf{y}}]-\mathbb{E}\left[f(\mathbf{x})\right]\mathbb{E}[\tilde{\mathbf{y}}] = 0\end{aligned}$$

Hence, all the cross terms in the expected value cancel out, and we're left with

$$\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] = \mathbb{E}\left[(f(\mathbf{x})-\mathbb{E}[\tilde{\mathbf{y}}])^2\right]+\mathbb{E}\left[(\mathbb{E}[\tilde{\mathbf{y}}]-\tilde{\mathbf{y}})^2\right]+\mathbb{E}\left[\boldsymbol{\epsilon}^2\right]$$

Using that $\mathbb{E}[\epsilon^2] = \sigma^2$ and writing the expected values as sums with the notation $f(\mathbf{x}_i) = f_i$, we get the desired expression. Since we have chosen $\mathbf{z}$ as our data variable we replace all the y variables with z, yielding

$$\mathbb{E}\left[(\mathbf{z}-\tilde{\mathbf{z}})^2\right] = \frac{1}{n}\sum_i (f_i-\mathbb{E}[\tilde{\mathbf{z}}])^2+\frac{1}{n}\sum_i(\tilde{z}_i-\mathbb{E}[\tilde{\mathbf{z}}])^2+\sigma^2 \tag{A1}$$

which is what we wanted to show.

Appendix B: Testing

In order to make sure our algorithms are running correctly, it is necessary to perform tests. We did this by comparing our results to those produced by scikit-learn. First of all we generated some simpler data for testing, namely an exponential:

$$f_{\text{Test}}(x) = exp(x)+\epsilon. \tag{B1}$$

Here ϵ denotes normally distributed noise, and x runs from $x_{\text{min}} = 0$ to $x_{\text{max}} = 1$ in $N = 50$ randomly distributed steps. This generates the testing data visualized in figure 4.

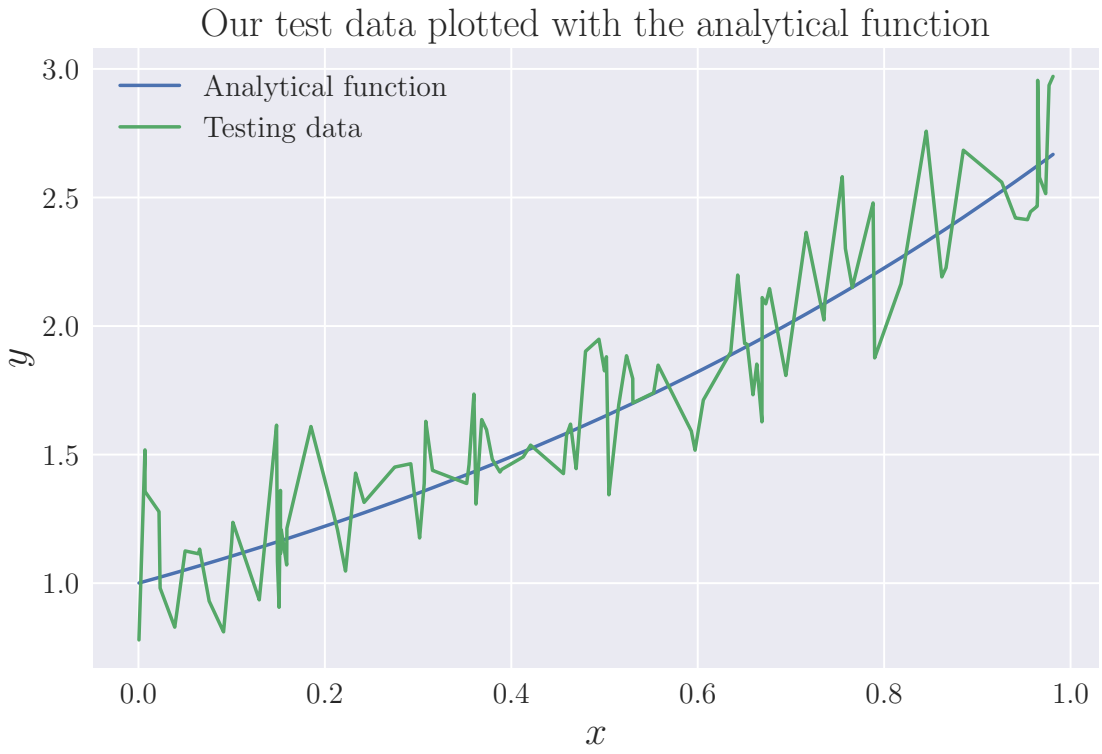

Figure 4. Here you we have plotted the testing data along with the analytical function.

First off we want to test the regression methods we have written.

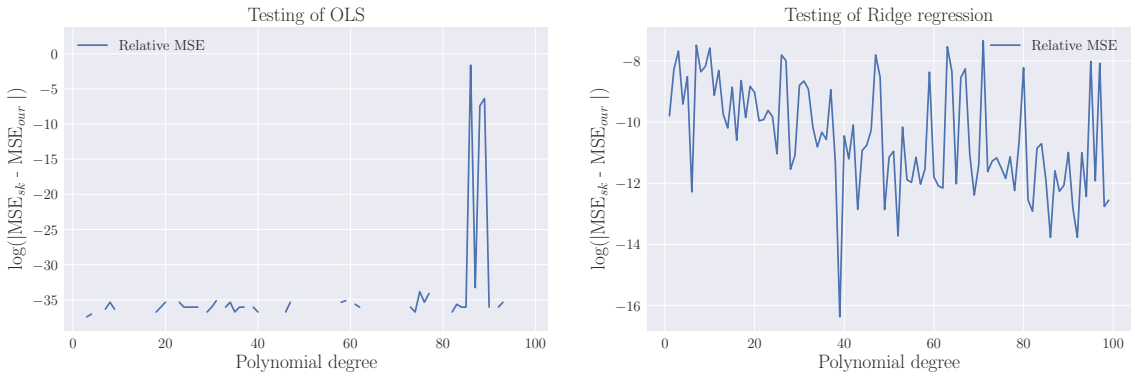

Figure 5. HEI EHIEHIAHFIASJDFKASDFJASDKF

[1] Ref.