Applying Regression and Resampling Techniques to Norwegian Terrain Data with Franke's Function as Test Function*

Erling Nupen
Ole Kristian Rustebakke
Brage Andreas Trefjord

October 9, 2023

Abstract

We used and tested different regression methods for fitting the two-dimensional Franke function and real data of norwegian terrain. The methods used were ordinary least squares (OLS), Ridge regression, and Lasso regression. We compared the mean square error (MSE) and the R^2-score for the different methods and for different degrees of the polynomials used in the fitting. From our results, it seems Lasso gives slightly worse results for the Franke function (large MSE and R^2 far from 1). OLS and Ridge are on the same scale. We studied the bias-variance trade-off for the OLS method, seeing that for low-degree polynomials the bias is high and the variance low, while for higher degrees the variance is high and the bias low. We also tested resampling methods like bootstrapping and cross-validation and saw how this would improve the MSE. The methods gave a bad fitting to the real-world terrain data. The MSE was high for all the methods, and it seems they are therefore not very applicable in this situation.

Contents

*GitHub Repository: `https://github.com/Bragit123/FYS-STK3155/tree/main/Project_1`

1 Introduction

In this project, we study different methods for performing linear regression on a data set. The methods we look at are the Ordinary Least Squares (OLS), Ridge regression, and Least Absolute Shrinkage and Selection Operator (LASSO) regression. To compare the different methods, we use them on the two-dimensional Franke function, and compute the mean square error (MSE) for the methods. We also look at resampling methods, namely bootstrapping and cross-validation to assess the performance of our models. After we have established our three methods of performing linear regression, we will apply them to real-world terrain data.

The linear model continues to be an important tool in statistics. Over the last 30 years, it has been exceptionally useful in analyzing data and understanding the relationship between input and output variables. In the linear model, the relationship between input and output variables is given in Eq (1)[1],

$$y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j, \tag{1}$$

where given an input X_j, we are able to determine the output y from the free parameters β_j, which are determined from the model fit to data. In this project we will look solely at polynomial fitting, which means that $X_j = x^j$ for some input variable x. In this case, Eq (1) is just

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p,$$

where p is the polynomial degree. Eq (1) can be written in a compact vector form, as represented in Eq (2),

$$\boldsymbol{y} = \boldsymbol{X}^T \boldsymbol{\beta}, \tag{2}$$

where the output $\boldsymbol{y}$ and the free parameters $\boldsymbol{\beta}$ are vectors, and the input $\boldsymbol{X}^T$ is a matrix, where each column corresponds to the polynomial degree, and each row corresponds to the different input values. $\boldsymbol{X}^T$ is commonly known as the feature matrix. Each column of the feature matrix contains its own predictor variable, which for this project would be the polynomial degree of the two input variables in the Franke-Function, Eq (16). The feature matrix used for fitting all three linear regression models to data is given in Figure 3.

The three methods of fitting a linear model to data, LASSO, OLS, and Ridge regression, all have their advantages and disadvantages. Both the advantages and disadvantages will be thoroughly explored, and an analysis of which method proved the most useful in fitting a linear model to the two-dimensional Franke function will be presented. In our analysis of the performance of our models, the two resampling techniques bootstrapping and cross-validation will be employed.

2 Theory

This chapter introduces the fundamental theory behind linear regression models, including OLS, Ridge regression, and LASSO regression. A comprehensive background of the resampling techniques is also presented.

2.1 The model

Consider the dataset $(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$, with independent variables $\boldsymbol{x}^T = [x_0, \ldots, x_{n-1}]$ and dependent variables $\boldsymbol{y}^T = [y_0, \ldots, y_{n-1}]$. We will assume that the data can be described by

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon},$$

where $\boldsymbol{f}$ is some continuous function, and $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ is a normally distributed error. We approximate the function $\boldsymbol{f}$ with a model $\tilde{\boldsymbol{y}}$. In this project the model will be a polynomial,

$$\tilde{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_i^p = \sum_{j=0}^{p} \beta_j x_i^j,$$

where $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_p]$ are the polynomial coefficients, and p is the degree of the polynomial. We can write this more compactly in vector form as

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

where $\boldsymbol{X}$ is the feature matrix,

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^p \\ 1 & x_1 & x_1^2 & \cdots & x_1^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^p \end{bmatrix}.$$

Finding a model $\tilde{\boldsymbol{y}}$ means finding the coefficients $\boldsymbol{\beta}$, and we do this by defining a cost function $C(\boldsymbol{\beta})$. The cost function should describe the error of our model, thus finding good coefficients means minimizing the cost function, i.e., finding $\boldsymbol{\beta}$ such that $\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$. It is this cost function that differs between our different models.

2.2 OLS

One way to measure the error in our model is to use the residual sum of squares (RSS), given in Eq (3)

$$\begin{aligned} \mathrm{RSS}(\boldsymbol{\beta}) &= \sum_{i=1}^{N} \left(y_i - \tilde{y}_i\right)^2, && (3) \\ &= \sum_{i=1}^{N} \left(y_i - \boldsymbol{X}_{i,*}\boldsymbol{\beta}\right)^2, && (4) \end{aligned}$$

where $\boldsymbol{X}_{i,*}$ refers to the i-th row of the feature matrix, and is summed over the columns. In vector notation, Eq (3) can be written as Eq (5),

$$\mathrm{RSS}(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}). \tag{5}$$

In ordinary least squares (OLS), the cost function is simply the RSS, i.e.,

$$C_{\mathrm{OLS}}(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}). \tag{6}$$

Differentiating the cost function we find that

$$\begin{aligned} \frac{\partial C_{\mathrm{OLS}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}). \end{aligned} \tag{7}$$

We minimize $C_{\mathrm{OLS}}(\boldsymbol{\beta})$ by setting Eq (7) equal to zero. This results in Eq (8) [1],

$$\begin{aligned} \frac{\partial C_{\mathrm{OLS}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= 0 \\ \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) &= 0 \\ \boldsymbol{\beta}_{\mathrm{OLS}} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \end{aligned} \tag{8}$$

We now have an expression for the coefficients $\boldsymbol{\beta}$, which we can use to compute our model $\tilde{\boldsymbol{y}}_{\text{OLS}} = \boldsymbol{X}\boldsymbol{\beta}_{\text{OLS}}$.

2.3 Ridge Regression

Ridge regression is part of a larger set of regression methods, known as Shrinkage methods. Ridge regression introduces a regularization term that penalizes the magnitudes of the free parameters. This term is known as the L2 penalty term, and its main function is to prevent overfitting and control the magnitude of the coefficients $\boldsymbol{\beta}$. This penalty term is introduced in the cost function as given in Eq (9),

$$
\begin{aligned}
C_{\text{Ridge}}(\boldsymbol{\beta}) &= \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2. \\
C_{\text{Ridge}}(\boldsymbol{\beta}) &= \left[(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right] + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}
\end{aligned} \tag{9}
$$

The free parameters in Ridge regression minimize a penalized RSS function. In Ridge regression, λ controls the degree of shrinkage applied to the free parameters. A higher value of λ leads to greater shrinkage, effectively reducing the magnitudes of the coefficients and preventing overfitting. If we minimize the cost function by setting its derivative equal to zero we get

$$
\boldsymbol{\beta}_{\text{Ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}. \tag{10}
$$

This gives us a different set of coefficients $\boldsymbol{\beta}_{\text{Ridge}}$, and thus also a different model $\tilde{\boldsymbol{y}}_{\text{Ridge}} = \boldsymbol{X}\boldsymbol{\beta}_{\text{Ridge}}$, than OLS.

2.4 LASSO

LASSO is similar to Ridge in that it is also a shrinkage method, however, it does have subtle and important differences. The L^2 ridge penalty term, which is defined by its L^2 squared norm, is now replaced by an L^1 norm, with the signature $\lambda \sum_{j=1}^{p} |\beta_j|$. The cost function for lasso regression is then

$$
\begin{aligned}
C_{\text{LASSO}}(\boldsymbol{\beta}) &= \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j. \\
C_{\text{LASSO}}(\boldsymbol{\beta}) &= \left[(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right] + \lambda ||\boldsymbol{\beta}||_1,
\end{aligned} \tag{11}
$$

where $||\cdot||_1$ is the L^1 norm, defined by $||\boldsymbol{v}||_1 = \sum_{i=1}^{n-1} |v_i|$ for some arbitrary vector $\boldsymbol{v}^T = [v_0, \ldots, v_{n-1}]$.

This new penalty term shrinks some free parameters to zero, essentially performing a feature selection, where irrelevant subsets are disregarded. This can be useful for datasets with irrelevant features[1]. Taking the derivative of the cost function gives us

$$\frac{\partial C_{\text{LASSO}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \, \text{sgn}(\boldsymbol{\beta}). \tag{12}$$

Unfortunately, setting Eq (12) equal to zero does not give us a nice analytical expression for $\boldsymbol{\beta}$ such as in Eq (8) and Eq (10). We can, however, use numerical methods to approximate $\boldsymbol{\beta}_{\text{LASSO}}$, which is what we do in this project when we use the functionality provided by the python package scikit-learn.

2.5 Accuracy of our model

There are two ways to determine the efficiency and accuracy of our model. The first is measuring the mean squared error (MSE) and another is the R^2 score. MSE and R^2 is defined as Eq (13)

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2, \quad R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2} \tag{13}$$

MSE is the squared error between the predicted output of our models and the observed values. R^2 is also known as the coefficient of determination[2], and is a measure of total variance accounted for by our model. The R^2 score assesses the proportion of the variance in our dependent variable explained by the independent variables in our regression model. It quantifies how well the model's predictions align with the actual data. It does this by comparing the sum of squared differences between the actual values and the predicted values (SSR) to the total sum of squares (SST). An R^2 score of 0 means that the model fails to explain any variance in the dependent variable. In this case, the model's predictions are no better than simply using the mean of the dependent variable as a constant prediction for all data points. It signifies that the model offers no improvement over the simplest baseline prediction.

The next section covers resampling techniques, which are ways to better analyze and understand which model has the best MSE. MSE will be used along with resampling techniques covered in the next section to asses which model is the best. R^2 will only be evaluated for the regular regression methods.

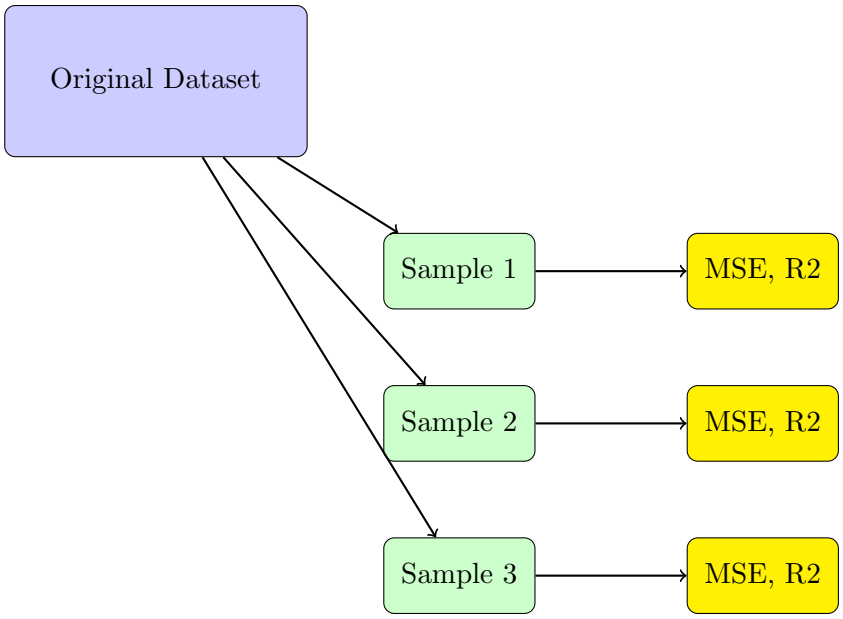


Figure 1: Illustration of Bootstrapping.

2.6 Resampling Techniques

In order to mitigate data imbalance within our datasets, we employ resampling techniques that result in the creation of more diverse subsets or sets for subsequent analysis. In these sections, we will further explain two primary methods of resampling the dataset, namely bootstrapping and cross-validation.

2.6.1 Bootstrapping

Bootstrap resampling generates new samples of identical size to the original dataset. This process entails repeatedly selecting data points from the original dataset, resulting in the inclusion of multiple instances of the original data within each newly created sample. The bootstrap technique is visualized in Fig 1

Bootstrapping is essential when it comes to Bias-Variance trade-off analysis. The repeated resampling of data helps produce several values for MSE and R^2, which potentially reduces the original bias of the original sample size. It also reduces variance, in that it allows us to discern how sensitive MSE and R^2 score is to sampling from the same data population.

2.6.2 Cross-validation

Cross-validation involves dividing our dataset into K equal-sized segments, often referred to as 'folds.' In each iteration of the process, one fold serves as a validation

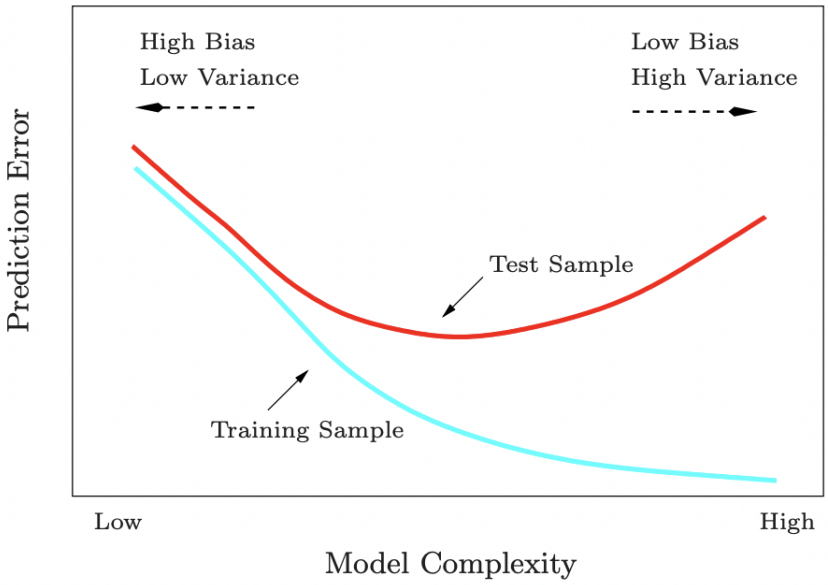


Figure 2: Figure from Hastie shows clearly how the relationship between bias and variance depends on model complexity.

set while the remaining folds act as a training set. This enables you to systematically assess the performance of your model by taking turns using each fold as a validation or test set for your model.

2.7 Bias-Variance Trade-Off

In machine learning, one central goal is to find a balance between bias and variance in your model. You want your model to be just complex enough to capture underlying patterns in your data, but not too complex, leaving your model sensitive to noise. Figure 2, borrowed from Hastie[1], is excellent in demonstrating the bias-variance trade-off.

Low-complexity models are equivalent in trying to fit potentially complex systems using an overly simplified mode. This could be trying to fit real-world data sets using first-order polynomials. The straight line must have high bias, as it is not able to capture the underlying pattern in the data. However, if you were to overcompensate by making your model overly complex, you end up capturing all the noise of the training data. With the overly complex model, you have essentially ended up fitting the noise, leading to a high variance, which leads to low model performance, on new unseen data.

The goal is therefore to choose a model, with just enough complexity to capture

underlying patterns in the data, while avoiding excessive complexity that might inadvertently capture noise.

We can use the cost function to obtain an expression for prediction error in terms of bias and variance. Expressing the cost function in terms of bias and variance is useful, as it exemplifies the relationship between the two. The cost function is a measure of the quality of our machine-learning model and can be expressed as Eq (14).

$$\begin{aligned} C(\boldsymbol{X}, \boldsymbol{\beta}) &= \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] \\ &= \mathbb{E}[\boldsymbol{y}^2] - 2\mathbb{E}[\boldsymbol{y}]\mathbb{E}[\tilde{\boldsymbol{y}}] + \mathbb{E}[\tilde{\boldsymbol{y}}^2]. \end{aligned} \tag{14}$$

Where $C(\boldsymbol{X}, \boldsymbol{\beta})$ is the cost function, $\boldsymbol{X}$ is the feature matrix representing our data and $\boldsymbol{\beta}$ represents the model parameters. Rewriting our cost function we obtain a relationship between bias and variance, Eq (15) known as bias-variance trade-off.

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \text{Bias}[f] + \text{var}[\tilde{y}] + \sigma^2. \tag{15}$$

Choosing the correct model complexity, in our case, the correct polynomial degree of our feature matrix, is discussed at length in Section 5.5, where Eq (15) will prove pivotal.

3 Method

3.1 Collecting Data

We will assume that our data points $\boldsymbol{y}$ can be described by a continuous $\boldsymbol{f}(\boldsymbol{x})$ and a normal distributed error $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$ as

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon}.$$

We will attempt to approximate $\boldsymbol{f}(\boldsymbol{x})$ with our model $\hat{\boldsymbol{y}}$, which we find using our regression methods. Our model will be of the form

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta},$$

For this project, our continuous function $\boldsymbol{f}(\boldsymbol{x})$, will be the well-known Franke function, given in Eq (16)

$$
\begin{aligned}
f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\
+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right).
\end{aligned} \tag{16}
$$

The Franke function takes in two input variables, x and y, defined on the interval $x, y \in [0,1]$. Therefore, our data is effectively scaled to fit within this specific domain.

The method we will first employ is OLS, described in Section 4.1, before trying Ridge and LASSO, described in section 2.3 and 5.4 respectively. We will construct our code first around uniformly distributed values of x and y on the given interval, trying to fit the Franke function for all three methods. Lastly, the Franke function will be replaced by real-world terrain data, where we will use the various re-sampling techniques in sections 2.6.1 and 2.6.2 to help us determine which of our models fit the data best.

The first step is to perform an OLS analysis on the Franke function, using a polynomial fit of the form

$$
\begin{aligned}
\tilde{\boldsymbol{z}} = \beta_{0,0} \\
&+ \beta_{1,0}x + \beta_{1,1}y \\
&+ \beta_{2,0}x^2 + \beta_{2,1}xy + \beta_{2,2}y^2 \\
&+ \beta_{3,0}x^3 + \beta_{3,1}x^2y + \beta_{3,2}xy^2 + \beta_{3,3}y^3 \\
&+ \cdots \\
&+ \beta_{n,0}x^n + \beta_{n,1}x^{n-1}y + \beta_{n,2}x^{n-2}y^2 + \cdots + \beta_{n,n-1}xy^{n-1} + \beta_{n,n}y^n \\
\tilde{\boldsymbol{z}} = \boldsymbol{X}\boldsymbol{\beta},
\end{aligned}
$$

where $\tilde{\boldsymbol{z}}$ is our model, $\boldsymbol{x}^T = [x_0, \ldots, x_n]$ and $\boldsymbol{y}^T = [y_0, \ldots, y_n]$ are the independent variables, $\boldsymbol{\beta}^T = [\beta_{0,0}, \beta_{1,0}, \beta_{1,1}, \ldots, \beta_{n,n-1}, \beta_{n,n}]$ are the coefficients and $\boldsymbol{X}$ is the feature matrix of out two-dimensional model. This feature matrix is shown in Figure 3. This design matrix is defined up to a degree of n, however, we will limit our highest polynomial degree to 20.

$$\begin{bmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 \\ \dots & x^n & x^{(n-1)}y & x^{(n-2)}y^2 & \dots & xy^{(n-1)} & y^n & & & \\ 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 \\ \dots & x^n & x^{(n-1)}y & x^{(n-2)}y^2 & \dots & xy^{(n-1)} & y^n & & & \\ 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 \\ \dots & x^n & x^{(n-1)}y & x^{(n-2)}y^2 & \dots & xy^{(n-1)} & y^n & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & & \end{bmatrix}$$

Figure 3: This is the x and y dependence of the polynomial fit of OLS, Ridge, and LASSO

We have opted for a train-test split of 80-20, meaning 80% of our data is used for training our model, whilst the remaining 20% is used as test data. A critical discussion of why an 80-20 split is optimal, is given in Section 5.1.

After splitting our data into a train and a test set, we apply our three methods of calculating the free parameters, β, on the training set, as outlined in sections 4.1, 2.3 and 5.4. Using the β coefficients, we are able to predict the output variable, as given in Eq (2). The model will be used to make predictions using both the training data, from which β is derived, and the test data, which would be entirely new for the model.

With the newly generated datapoints, $\tilde{z}_{\text{train}}$, and $\tilde{z}_{\text{test}}$, both MSE and R^2 score will be calculated and compared. For the resampling methods, only MSE will be considered.

For the parametrization of real-world terrain data, the procedure just outlined will be repeated, and cross-validation will be used to determine which linear regression technique, OLS, ridge, or LASSO, is optimal in the parametrization of real-world terrain data.

We can calculate the expectation value $\mathbb{E}(y_i)$ and variance $\text{Var}(y_i)$ for an element y_i of $\boldsymbol{y}$. We then find the expectation value to be

$$\mathbb{E}(y_i) = \sum_j x_{ij}\beta_j = \boldsymbol{X}_{i,*}\boldsymbol{\beta},$$

where $\boldsymbol{X}_{i,*}$ is the i-th row of the feature matrix summed over every column. The variance is

$$\mathrm{Var}(y_i) = \sigma^2.$$

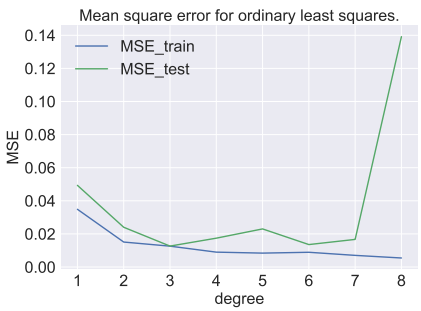
A complete derivation of these expressions can be found in Appendix A.

4 Results

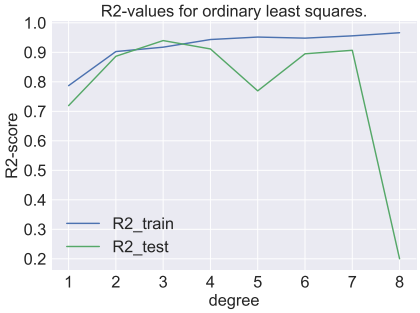
4.1 OLS

The MSE error for the OLS method of regression used on the Franke function with an added stochastic noise using the normal distribution $N(0, 1)$ is shown in Figure 4a. Here we have tested for polynomials from degree 1 to 8. 100 data points were used.

Figure 4b shows the R^2-score for the OLS method on the Franke function with the same conditions.



(a) MSE-error for OLS method, for different polynomial degrees.



(b) R^2-score for OLS, for different polynomial degrees.

Figure 4: Error plots of ordinary least squares fitting to the Franke function for both training and test data.

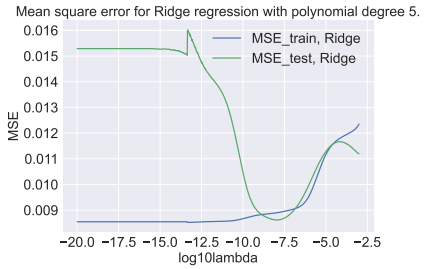
4.2 Ridge

The MSE error for the Ridge method of regression is shown in figure 5a. This is for the Franke function with the added stochastic noise. We used a degree 5 polynomial, and λ-values ranging from 10^{-20} to 10^{-3}. 100 data points were used.

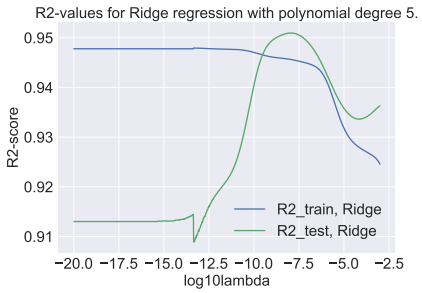
Figure 5a shows the R^2-score for the Ridge method of regression used on the Franke function with the same conditions.

4.3 Lasso

The MSE error for the Lasso method of regression is shown in Figure 6a. Lasso is used on the Franke function with the added stochastic noise. We used a degree



(a) MSE-error for Ridge method with different values of the λ-parameter, the degree of the polynomial is 5

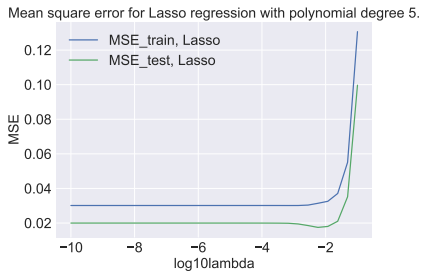


(b) R^2-score for Ridge with different values of the λ-parameter, the degree of the polynomial is 5

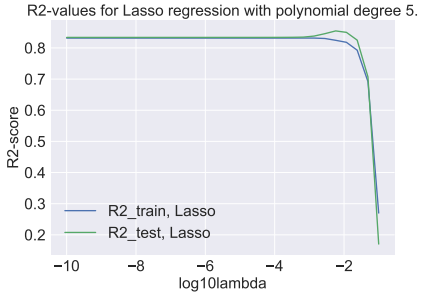
Figure 5: Error plots of ridge regression fitting to the Franke function for both training and test data.

5 polynomial, and λ-values ranging from 10^{-10} to 10^{-1}. 100 data points were used.

Figure 6b shows the R^2-score for the Lasso method of regression used on the Franke function with the same conditions as was used in Figure 6a.



(a) MSE-error for Lasso method with different values of the λ-parameter, the degree of the polynomial is 5.



(b) R^2-score for Lasso with different values of the λ-parameter, the degree of the polynomial is 5.

Figure 6: Error plots of Lasso regression fitting to the Franke function for both training and test data.

4.4 Bootstrapping and crossvalidation

The MSE error for the OLS method of regression compared with bootstrapping on this method is shown in Figure 7. Again we use the Franke function with an added stochastic noise. 10 bootstrap samples were used, and the polynomials range from degree 1 to 8. 100 data points were used.

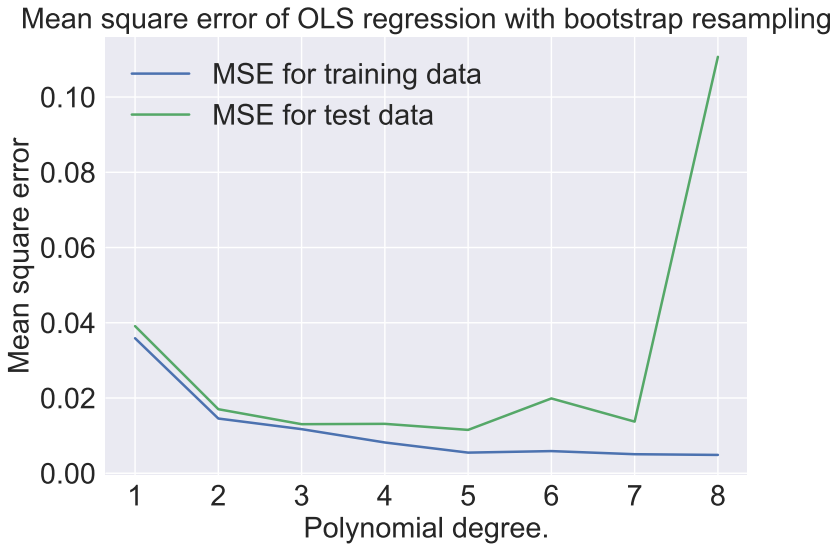


Figure 7: MSE-values after using Bootstrapping method on OLS for different polynomial degrees.

We use the cross-validation resampling method, and compare it to the normal way of applying OLS to find the MSE which is shown in figure 8a. The Franke function with the added stochastic noise is still what we assess. 5 folds were used and the polynomials ranged from degree 1 to 8. 100 data points were used.

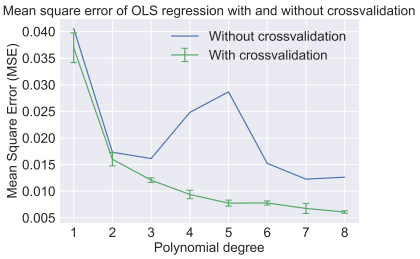
The MSE of Ridge compared with when we use cross-validation is shown in Figure 8b. Lastly, we use cross-validation and compare it to the MSE of the normal Lasso method and plot it in Figure 8c. Stochastic noise is added to the Franke function. 5 folds were used and the polynomial is of degree 5, λ ranges from 10^{-10} to 10^{-3}. 100 data points were used.

4.5 Terrain data

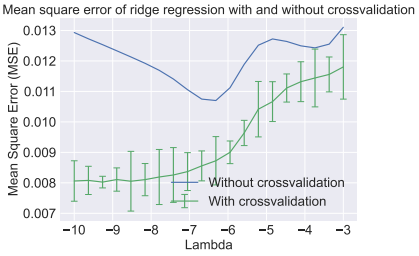
An aerial picture of the terrain we want to fit is shown in figure 10a. We will use OLS, Ridge, and Lasso on this data with cross-validation as the resampling method.

Figure 9a shows the MSE error for the OLS method of regression used on a portion of the terrain data corresponding to the 100 first data points in the x and the y direction (10000 points in total). We also use the resampling method called cross-validation and compare it to the result we get for applying the OLS method normally. 5 folds were used and the polynomial ranges from degree 2 to 8.

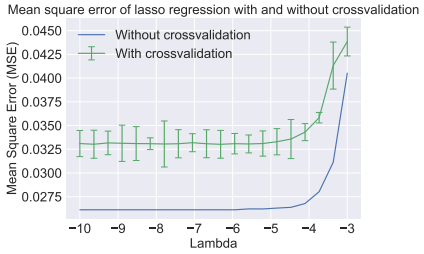
In Figure 9b we can see the MSE error for the Ridge method of regression with



(a) MSE for cross-validation compared with no resampling using OLS, we use 5 folds

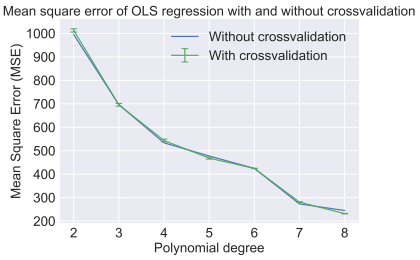


(b) MSE for cross-validation compared with no resampling using Ridge, the degree is 5 and we use 5 folds

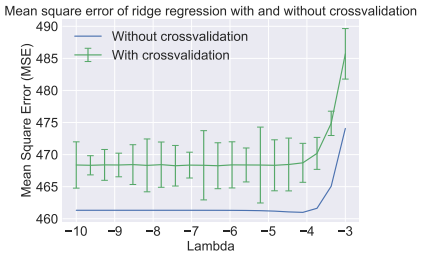


(c) MSE for crossvalidation compared with no resampling using Lasso, the degree is 5 and we use 5 folds

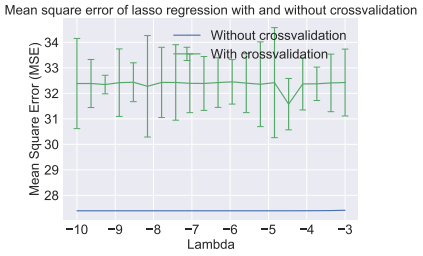
Figure 8: Mean square error (MSE) for the models to the Franke function with and without cross validation.



(a) Crossvalidation compared to no resampling on the terrain data for the OLS method



(b) Cross-validation compared to no resampling on the terrain data for the Ridge method, the degree is 5



(c) Cross-validation compared to no resampling on the terrain data for the Lasso method, the degree is 5

Figure 9: Mean square error (MSE) for the models to the terrain data with and without cross validation.

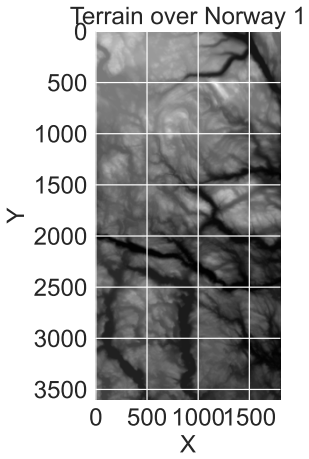
and without cross-validation. The polynomial is of degree 5, λ range from 10^{-10} to 10^{-3}.

We also use the Lasso method of regression on the terrain and plot the MSE error with and without cross-validation in Figure 9c. This corresponds to the 20 first data points in the x and the y direction (400 points in total). 5 folds were used and the polynomial is of degree 5, λ range from 10^{-10} to 10^{-3}.

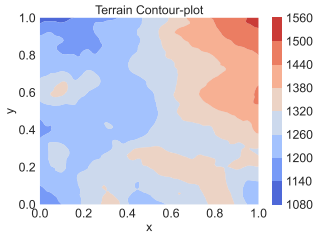
A contour plot of the first 100 data points in the x and y directions is shown in Figure 10b. We want to compare this with the OLS and Ridge results.

Figure 10c shows a contour plot of the terrain data of the first 20 data points in the x and y directions, to be compared with Lasso results.

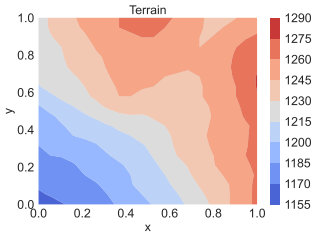
In comparison we have in figure 11a, 11b and 11c contour plots of the fitted terrain data using respectively the OLS, Ridge, and the Lasso method for regression. We found the degree that gives the minimum MSE error in OLS to be 8 and used this. The optimal λ was respectively $\lambda = 10^{-10}$ and $\lambda = 0.001$ for Ridge and Lasso, and we still use degree 5.



(a) Image of the terrain we want to describe using a two-dimensional polynomial. The terrain is from a region close to Stavanger in Norway.

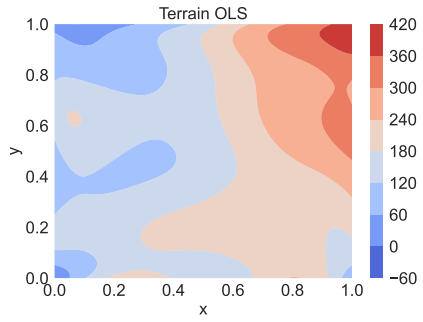


(b) A contour plot of the terrain data of the first 100 data points in the x and y direction.

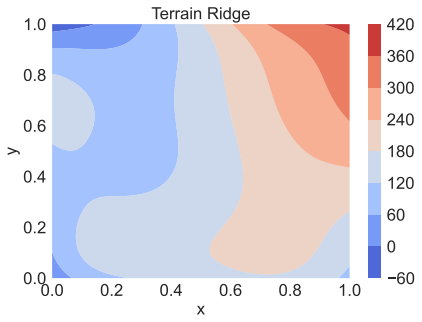


(c) A contour plot of the terrain data of the first 20 data points in the x and y direction.

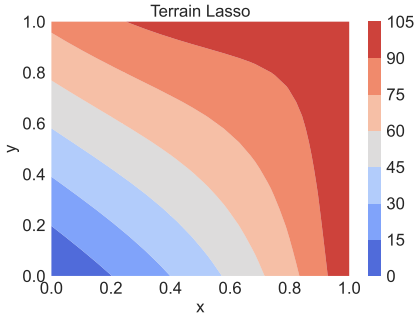
Figure 10: Contour plots of the terrain data, both of the whole data set, and for the different subsets of the data that was used in our code.



(a) A contour plot of the fitted terrain data using the OLS method with degree 8. The model was used on a subset of the terrain data, as showed in figure 10b.



(b) A contour plot of the fitted terrain data using the Ridge method with degree 5 and $\lambda = 10^{-10}$. The model was used on a subset of the terrain data, as showed in figure 10b.



(c) A contour plot of the fitted terrain data using the Lasso method with degree 5 and 0.001. The model was used on a subset of the terrain data, as showed in figure 10c.

Figure 11: Contour plots of the fitted terrain data from the different regression models.

5 Discussion

In this section, we conduct an analysis of the figures presented in Section 4. Our aim is to provide a critical examination of how the various metrics, indicative of model quality, evolve with changes in the polynomial degree. This discussion sheds light on the potential impact of increasing complexity on our model's predictive performance. We will also discuss various choices made throughout this analysis, such as how we decided on an 80-20 train-test split. Additionally, we will determine which model best captures the characteristics of Norwegian terrain data, and whether polynomial regression is a good way to describe such real-life input.

5.1 Why 80-20 test train split

80-20 split between the train and test data is common in the field of machine learning, however, there seems to be no general consensus around it. It stems from the well-known Pareto principle[1] [3], authors such as Picard and Beck have recommended that the testing set be 25% - 50% of the data[4], whilst Afendras and Markatou recommend 50%[5]. We opted for an 80-20 split, as we wanted to allocate a large chunk of data to optimize the training of our model. Also due to our data size, a 20% test set is sufficient. Especially for the slower Lasso method with a small dataset more data should be allocated to training.

5.2 OLS

Figure 4a shows a slight decrease in MSE as we increase the polynomial degree from 1 to 2, for both the test and train datasets. From degree 2 to 7, the error remains relatively constant for both test and train data. At degree 8, the error of the test dataset increases significantly. This is likely due to a high variance (although the bias might be low), leading to our model overfitting the test data. The reason for this overfitting is that a large order polynomial oscillates a lot, but has a degree large enough to fit the training data points. It describes only the given data points, and does not predict the future ones very well.

Figure 4b shows an increasing R^2 score from degree 1 to 2 (the closer R^2-score is to 1 the better it is), before leveling off from degree 2 to 7 for both the training and test data sets. At degree 8, the R^2 score for the test data plummets. This sudden decrease in R^2 score coincides with an extreme increase of the MSE, suggesting our model complexity is too high, leading to an overfitting of data. Overfitting the data erases the independent variables and the dependent variables' relationship, resulting in a very poor R^2 score.

[1]The Pareto Principle is an interesting rabbit hole to go down, however, it is not elaborated upon further in this project.

5.3 Ridge

Figure 5a shows a wide split between the MSE of the test and training data for a polynomial fit of degree 5, plotted against the common logarithm of regularization parameter λ up to $log10(\lambda) = -13$. After $log10(\lambda) = -13$, the error rapidly decreases, before aligning with the error of the training set. From about $log10(\lambda) = -9$ up to $log10(\lambda) = -4$ the error in both sets remains of the size. After $log10(\lambda) = -4$, the training set performs worse than the test set and has a larger MSE. The initial decrease in the MSE for the test set is due to the regularization parameters preventing overfitting, which improves the model's ability to generalize to the unseen test data. The MSE is also of a smaller scale than the 5th-degree OLS method, which gives an MSE error of around 0.2. However, as the λ's increases the β parameters become more and more suppressed, which in terms underfits our model to test data. This underfitting reduces the model's ability to capture complex relationships within our data, resulting in a higher MSE for both test and training data. It seems like for $log10(\lambda) = -4$ the MSE improves again for the test data however this is likely not a trend, and if we were to look at a larger interval of λ values, the MSE's would both significantly increase.

Figure 5b is almost the inverse of Figure 5a. MSE and R^2 score exhibit an inverse-like relationship. This can be explained by the model improving, leading to a better R^2 score which explains a larger portion of the variance in the dependent variable, resulting in a smaller MSE. Conversely, when the R^2 score fails to explain the variance in the dependent variable, we see the MSE rise and the predictive capabilities of our models become poorer.

5.4 LASSO

Figure 6a shows a constant MSE for both the training and test set from a $log10(\lambda) = -10$, to $log10(\lambda) = -2$, at which point the error exponentially increases. Here λ is clearly too big, punishing not just big oscillations but smaller ones as well.

The R^2-score in Figure 6b illustrates the same points that the MSE error plot does, since it is basically the inverted version of this plot.

Compared to the other models the behaviour for Lasso seems weird. The MSE is larger for the train than for the test data which is not what we expect since the train data is what we use to find the model. The MSE error is also of a larger scale than for OLS and Ridge. The reason why could be because Lasso is better with problems that contain a large number of variables but few observations [1]. Here we only have x and y as the variables, but some systems in machine learning might need many more.

5.5 Model Complexity (Bias-variance trade-off)

Figure 7 shows the bias-variance trade off that occurs when the degree increases. As the degree increases the bias will be lower as the mean centers around our actual input data, but variance increases when we get overfitting. For low degrees, the bias is higher, because of underfitting the prediction is far away from where the data lies, but the variance is lower because the model is simpler and we have lower degree terms. Higher degree terms generally increase or decrease rapidly, hence the high variance. Using bootstrap for resampling assures us that it indeed is overfitting that happens here, since we take the average of MSE for different datasets (same data in total but combined differently).

5.6 Cross-validation

The cross-validation is illustrated for OLS, Ridge, and Lasso in Figures 8a, 8b and 8c respectively. Especially for OLS and Ridge, we can see that taking the average when dividing the test and splitting in different ways evens out the plot and gets rid of sudden spikes in MSE. This way we can see that the results we get actually depend on what we use as a test and what we use as train data. Although in this case these spikes can be statistical deviations because the error bars for both OLS and Ridge seem to be of a small scale, which means that we could have split the train and test data differently and still gotten similar results. For Lasso, the MSE is actually larger with cross-validation which could be for the same reasons discussed in 5.4.

5.7 Terrain

The MSE error for all the methods is quite large here, as seen in Figures 9a, 9b, 9c, and this is because the terrain we want to analyze is hilly and will be hard to approximate mathematically. We see from the contour plots (Figures 11a and 11b compared to 10b, and Figure 11c compared to 10c) that the general shape has some similarities in real life and for our mathematical approximations, but OLS, Ridge and Lasso predicts nature to be way smoother than it is in our varied Norway. It is also noteworthy that the terrain we analyzed is only a small portion of the terrain in Figure 10a. A fit of the whole data set would probably be even worse due to the fluctuations in the terrain increasing for larger subsets.

5.8 Further notes

Notice that 100 data points for OLS and Ridge can seem like a small amount. The reason why we use this is both to see the bias-variance trade-off more easily in the OLS method, and because for more than 100 data points in the Lasso method, it can run quite slowly. Therefore we stick to 100 so we can compare the methods more easily.

6 Conclusion

OLS and Ridge seem to be the best models for the Franke function, while Lasso gives a larger MSE error. By using bootstrapping for higher degrees of the fit in OLS we see the expected bias-variance trade off with a low bias but high variance for larger degree polynomials. Cross-validation is also used as a resampling method and shows us that the methods' quality depends on how we split the dataset. Although, the deviation in the MSE is not very large for OLS and Ridge in different splits, so therefore the methods are well applied. The terrain data is not well fitted using any data, because it is way to varied.

References

[1] J. F. Trevor Hastie Robert Tibshirani, *The elements of statistical learning*. Springer, 2009.

[2] R. A. Fisher, "Xv.—the correlation between relatives on the supposition of mendelian inheritance.," *Earth and Environmental Science Transactions of The Royal Society of Edinburgh*, vol. 52, no. 2, 399–433, 1919. DOI: 10.1017/S0080456800012163.

[3] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, pp. 531–538, 2022. DOI: 10.1002/sam.11583. [Online]. Available: https://doi.org/10.1002%2Fsam.11583.

[4] R. R. Picard and K. N. Berk, "Data splitting," *The American Statistician*, vol. 44, no. 2, pp. 140–147, 1990, ISSN: 00031305. [Online]. Available: http://www.jstor.org/stable/2684155 (visited on 10/04/2023).

[5] G. Afendras and M. Markatou, *Optimality of training/test size and resampling effectiveness of cross-validation estimators of the generalization error*, 2015. arXiv: 1511.02980 [math.ST].

A Appendix: Deriving statistical quantities

A.1 Assumptions

We will assume that our data can be described by some continuous function $\boldsymbol{f}(\boldsymbol{x})$, together with a normally distributed error $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$. That is, we assume that our data $\boldsymbol{y}$ is

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon}.$$

We then approximate the function $\boldsymbol{f}(\boldsymbol{x})$ with our model $\tilde{\boldsymbol{y}}$

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

where $\boldsymbol{X}$ is the feature matrix, and $\boldsymbol{\beta}$ contains the coefficients of our model.

We will use this assumption to derive expressions for the expectation value and variance of y_i and $\hat{\boldsymbol{\beta}}$, starting with y_i.

A.2 The expectation value of the model

The expectation value of y_i is

$$\begin{aligned}\mathbb{E}(y_i) &= \mathbb{E}\left(\sum_i x_{ij}\beta_j + \varepsilon_i\right) \\ &= \mathbb{E}\left(\sum_i x_{ij}\beta_j\right) + \mathbb{E}(\varepsilon_i).\end{aligned}$$

Here x_{ij} and β_j are both non-stochastic variables, thus $\mathbb{E}(\sum_j x_{ij}\beta_j) = \sum_j x_{ij}\beta_j$. Since ε follows a normal distribution, we have that $\mathbb{E}(\varepsilon_i) = 0$. This gives us

$$\mathbb{E}(y_i) = \sum_i x_{ij}\beta_j \tag{17}$$

$$\mathbb{E}(y_i) = \boldsymbol{X}_{\boldsymbol{i},*}\boldsymbol{\beta}, \tag{18}$$

where $\boldsymbol{X}_{\boldsymbol{i},*}$ means that we look at row i of $\boldsymbol{X}$, and sum over all columns.

A.3 The variance of the model

The variance of y_i is

$$\begin{aligned}
\mathrm{Var}(y_i) &= \mathbb{E}\left([y_i - \mathbb{E}(y_i)]^2\right) \\
&= \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \\
&= \mathbb{E}\left[(\boldsymbol{X_{i,*}}\boldsymbol{\beta} + \varepsilon_i)^2\right] - (\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2 \\
&= \mathbb{E}\left[(\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2 + 2\varepsilon_i \boldsymbol{X_{i,*}}\boldsymbol{\beta} + \varepsilon_i^2\right] - (\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2 \\
&= \mathbb{E}\left[(\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2\right] + \mathbb{E}\left[2\varepsilon_i \boldsymbol{X_{i,*}}\boldsymbol{\beta}\right] + \mathbb{E}\left[\varepsilon_i^2\right] - (\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2.
\end{aligned}$$

Since $\boldsymbol{X_{i,*}}$ and $\boldsymbol{\beta}$ are both non-stochastic variables we have that

$$\mathbb{E}\left[(\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2\right] = (\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2,$$

and

$$\mathbb{E}\left[2\varepsilon_i \boldsymbol{X_{i,*}}\boldsymbol{\beta}\right] = 2\boldsymbol{X_{i,*}}\boldsymbol{\beta}\mathbb{E}[\varepsilon_i] = 0,$$

where the last equality comes from the normal distribution of $\boldsymbol{\varepsilon}$. In addition, since $\boldsymbol{\varepsilon}$ follows a normal distribution, the expectation value of ε_i^2 is just its standard deviation, $\mathbb{E}(\varepsilon_i^2) = \sigma^2$. Combining these expressions, we get

$$\begin{aligned}
\mathrm{Var}(y_i) &= \cancel{(\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2} + \sigma^2 - \cancel{(\boldsymbol{X_{i,*}}\boldsymbol{\beta})^2} \\
\mathrm{Var}(y_i) &= \sigma^2.
\end{aligned}$$

A.4 The expectation value of the coefficients

The ordinary least squares expression for the coefficients $\hat{\boldsymbol{\beta}}$ is

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

Using this we find that the expectation value of $\hat{\boldsymbol{\beta}}$ is

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

Since $\boldsymbol{X}$ is non-stochastic, so is $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T$, and we can separate it from the expectation value in our expression. Using this we get

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\mathbb{E}\left[\boldsymbol{y}\right]. \\
&= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \boldsymbol{\beta},
\end{aligned}$$

where we have used that $\mathbb{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta}$, which follows directly from $\mathbb{E}(y_i) = \boldsymbol{X}_{\boldsymbol{i},*}\boldsymbol{\beta}$.

A.5 The variance of the coefficients

The variance of $\hat{\boldsymbol{\beta}}$ is

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left[(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta}))^2\right] \\
&= \mathbb{E}\left[(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta}))(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta}))^T\right].
\end{aligned}$$

Again, we use that $\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ to get

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left(\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \beta)\right]\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \beta)\right]^T\right) \\
&= \mathbb{E}\left(\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \beta)\right]\left[\boldsymbol{y}^T\boldsymbol{X}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\right)^T - \boldsymbol{\beta}^T\right]\right) \\
&= \mathbb{E}\left(\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \beta)\right]\left[\boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}^T\right]\right) \\
&= \mathbb{E}\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right. \\
&\qquad \left. -(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\boldsymbol{\beta}^T - \boldsymbol{\beta}\boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^T\right] \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \\
&\qquad - (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y})\boldsymbol{\beta}^T - \boldsymbol{\beta}\mathbb{E}(\boldsymbol{y}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^T.
\end{aligned}$$

We have that $\mathbb{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta}$, thus $\mathbb{E}(\boldsymbol{y}^T) = (\boldsymbol{X}\boldsymbol{\beta})^T = \boldsymbol{\beta}^T\boldsymbol{X}^T$. We also have that

$$\begin{aligned}
\mathrm{Var}(\boldsymbol{y}) &= \mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T) - \mathbb{E}(\boldsymbol{y})\mathbb{E}(\boldsymbol{y}^T) \\
\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T) &= \mathbb{E}(\boldsymbol{y})\mathbb{E}(\boldsymbol{y}^T) + \mathrm{Var}(\boldsymbol{y}) \\
\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T) &= \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2.
\end{aligned}$$

Putting these expressions together, we have

$$
\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \\
&\qquad - \cancel{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}}\boldsymbol{\beta}\boldsymbol{\beta}^T - \boldsymbol{\beta}\boldsymbol{\beta}^T\cancel{\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}} + \boldsymbol{\beta}\boldsymbol{\beta}^T. \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\
&= \cancel{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}}\boldsymbol{\beta}\boldsymbol{\beta}^T\cancel{\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}} \\
&\qquad + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\sigma^2\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\
&= \cancel{\boldsymbol{\beta}\boldsymbol{\beta}^T} + \cancel{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}}\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \cancel{\boldsymbol{\beta}\boldsymbol{\beta}^T} \\
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}.
\end{aligned}
$$

B Appendix: MSE in terms of bias and variance

We will assume, as above, that our data can be described as

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\varepsilon},$$

where $\boldsymbol{f}$ is a continuous function, and $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ is a normally distributed error. Let $\tilde{\boldsymbol{y}}$ be the model approximating $\boldsymbol{f}$. Then the variance of $\tilde{\boldsymbol{y}}$ is

$$\mathrm{Var}[\tilde{\boldsymbol{y}}] = \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\right],$$

and the bias of $\tilde{\boldsymbol{y}}$ is

$$\mathrm{Bias}[\tilde{\boldsymbol{y}}] = \mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\right].$$

We can find an expression for the mean square error $\mathrm{MSE}(\tilde{\boldsymbol{y}}) = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]$ of our model:

$$
\begin{aligned}
\mathrm{MSE}(\tilde{\boldsymbol{y}}) &= \mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] \\
&= \mathbb{E}\left[(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{\varepsilon}+\tilde{\boldsymbol{y}})^2\right] \\
&= \mathbb{E}\left[(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{\varepsilon})^2 - 2(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{\varepsilon})\cdot\tilde{\boldsymbol{y}}+\tilde{\boldsymbol{y}}^2\right] \\
&= \mathbb{E}\left[(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{\varepsilon})^2\right] - \mathbb{E}\left[2(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{\varepsilon})\cdot\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}^2\right] \\
&= \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})^2 + 2\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{\varepsilon}+\boldsymbol{\varepsilon}^2\right] \\
&\qquad - 2\mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})\cdot\tilde{\boldsymbol{y}}+\boldsymbol{\varepsilon}\cdot\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}^2\right] - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]^2 + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]^2 \\
&= \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})^2\right] + \cancelto{0}{2\mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{\varepsilon}\right]} + \cancelto{\sigma^2}{\mathbb{E}\left[\boldsymbol{\varepsilon}^2\right]} \\
&\qquad - 2\mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})\cdot\tilde{\boldsymbol{y}}\right] - \cancelto{0}{2\mathbb{E}\left[\boldsymbol{\varepsilon}\cdot\tilde{\boldsymbol{y}}\right]} + \mathrm{Var}[\tilde{\boldsymbol{y}}] + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]^2 \\
&= \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})^2\right] - 2\mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x})\right]\cdot\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]^2 + \mathrm{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 \\
&= \mathbb{E}\left[(\boldsymbol{f}(\boldsymbol{x})-\mathbb{E}[\tilde{\boldsymbol{y}}])^2\right] + \mathrm{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 \\
\mathrm{MSE}(\tilde{\boldsymbol{y}}) &= \mathrm{Bias}[\tilde{\boldsymbol{y}}] + \mathrm{Var}[\tilde{\boldsymbol{y}}] + \sigma^2.
\end{aligned}
$$

Here we can see that the mean square error, and therefore also the cost function of the ordinary least squares method (see Eq (6)), is the sum of the bias and variance of our model, and the unchanging systematic error σ^2 from our data.