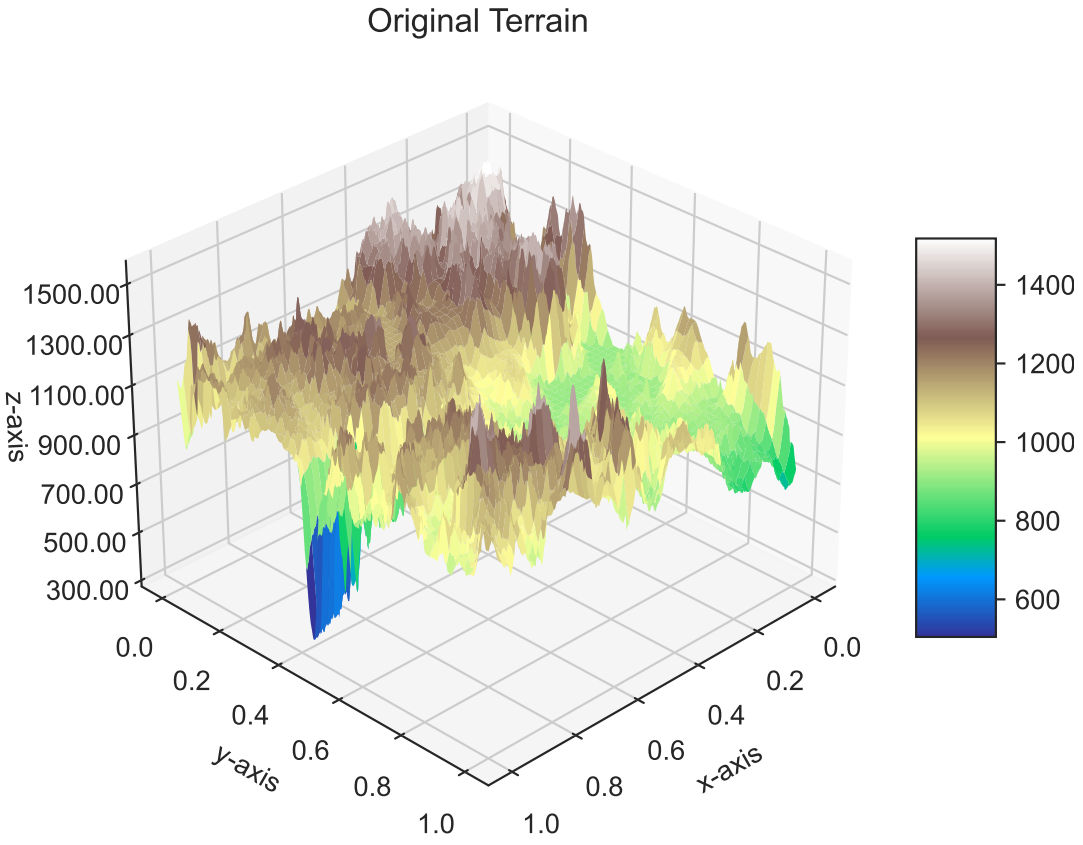
Linear Regression of Franke Function and Topographical Data

FYS-STK4155

Erlend Lemva Ousdal,
Kristen Joy Valseth,
Vilde Gahr Sturtzel Lunde



Faculty of Natural Sciences
University of Oslo
October 2023

Linear Regression of Franke Function and Topographical Data

Erlend Lemva Ousdal, Kristen Joy Valseth, Vilde Gahr Sturtzel Lunde

University of Oslo

Abstract

In this study, machine learning was employed to fit the Franke function and real topographical data to polynomial functions using linear regression models, aiming for precise terrain modeling. Accurate models of topographical data are relevant for applications such as urban planning and development, and geological investigation. For these purposes, model accuracy is essential. In order to optimize the models in this work, the resampling technique, polynomial degree for the fit, regularization factors, and the bias-variance trade-off were optimized for three different linear regression models.

For the Franke function, the performance of the model in this project improved greatly with increasing polynomial degrees up to the fifth degree for both ordinary least squares, Ridge, and Lasso regression techniques. However, subsequent improvement was negligible for further increasing the polynomial degree. The ordinary least squares model outperformed the Ridge and Lasso models when both the error of the fits and their computational powers were taken into consideration. The optimal fit was provided using ordinary least squares with 10-fold cross-validation and a fifth-degree polynomial, obtaining a mean squared error of 0.00130, which is 0.3 % of the mean value of the Franke function. For the terrain data, the same ordinary least squares model resulted in the best fit. However, the error of the fit was much higher than for the Franke function, having a mean squared error of 8138, which is 747 % of the mean value of our terrain data. This was assumed to be due to the increased complexity of the data to be fitted.

In conclusion, the Franke function was successfully fitted using ordinary least squares with a fifth-degree polynomial, while the fit was quite inaccurate for the terrain data. Further work could explore the implementation of alternative functions than polynomials for the fitting. Additionally, the obtained models should be tested on diverse terrains to explore the transferability to various topographical data sets to investigate the versatility of the models.

All code developed for this project, along with additional files and figures are provided at:
https://github.com/krisvals/FYS4155_Projects/tree/main/Project1.

1 Introduction

Fitting data sets using linear regression can be advantageous to gain simple and interpretable models which can be used to explore features in the data. Topographical data can be relevant to model a wide range of technical applications [1]. In the context of terrain data, the resulting models can be used for everything from urban planning and development of infrastructure to geological investigations. Polynomial functions can be promising for fitting topographical data since they are easy to customize and implement using standard tools. Challenges related to fitting topographical data using linear regression concern the balance between sufficient accuracy of the predictions and the simplicity of the model [2]. These challenges will be addressed in this work.

In this study, the Franke function and topographical data from a landscape in Norway were fitted using polynomial functions. Three different linear regression models were used and compared, namely ordinary least squares (OLS), Ridge, and Lasso. The models were evaluated based on the mean square error (MSE), the coefficient of determination (R^2), the bias, and the variance of the fit. Additionally, the resampling techniques cross-validation and bootstrapping were employed to enhance the result.

In the Theory and Method part of this work, the theory behind the model development will be presented, and the choices made throughout the development will be discussed, including the choice of parameters and the optimization techniques used on the models. In the Result and Discussion section, the results from the fitting of the Franke function and the terrain data will be presented and discussed. Finally, in the Conclusion and Future Work section, the conclusions obtained in this work will be summarized, and possible further work for optimizing the fits will be proposed.

2 Theory and Method

This section introduces the theoretical foundation needed to understand the work performed in this project, along with a description of the method used for model development. In this work, Python was used as a coding language as it has become the standard programming language in many data science applications [3]. We developed our own code, using machine learning algorithms to implement

linear regression models used for fitting the data. The results from the produced code were compared with the results obtained using the Python machine learning library scikit-learn (`sklearn`) [4]. The `sklearn` library contains state-of-the-art machine learning algorithms, and is open source, meaning the algorithms in this work can be easily reproduced by other groups [3].

The code has been implemented using Jupyter Notebooks, which has been saved as `.ipynb` files in Github so the readers of this work can run the code themselves. The code is divided into two different files: `Project1.ipynb` contains the code for the model optimization, while `Project1_Illustrations.ipynb` contains the code for illustrating the fits performed by the optimized models. The method of development for the code in these files will now be described.

2.1 Input Data

The Franke function and real topographical data were fitted in this project. This data was visualized through 3D plots provided by the `matplotlib` library for data inspection and to visually compare the results obtained from the regression models to the original data [5].

2.1.1 Franke Function

The Franke function consists of two Gaussian peaks and is often used as a test function in machine learning to approximate data. The function is defined as [6]:

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

where x and y are coordinates in a plane. The Franke function was used for x and y [0,1], where 20 steps were taken for each parameter inside the interval. The Franke function is illustrated graphically in Figure 1. As seen, the figure had both peaks and valleys, resembling smooth terrain.

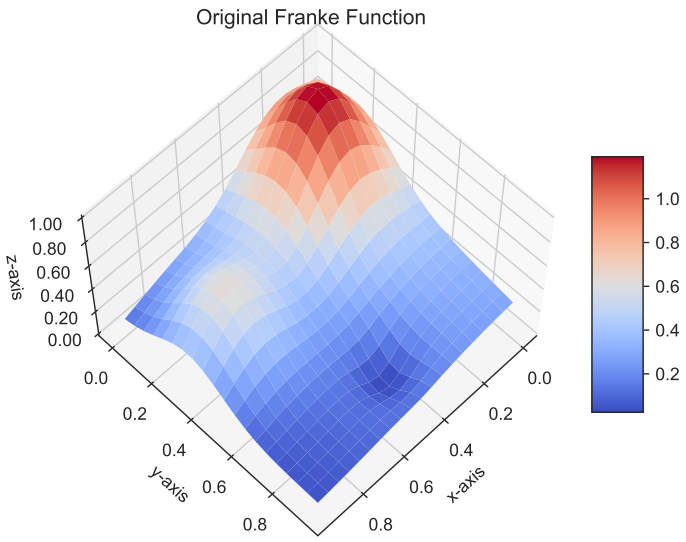


Figure 1: Franke function, defined in Equation 1.

Stochastic noise was added to the Franke function, as $f(x,y,\sigma) = f(x,y) + N(0,1)$, where $f(x,y)$ is the Franke function and $N(0,1)$ is the added noise, which has a normal distribution. The Franke function with the added noise is illustrated in Figure 2. The main features from Figure 1 were still visible, but the surface was no longer smooth and there were more fine features. This figure resembles a rougher terrain than what the original Franke function did.

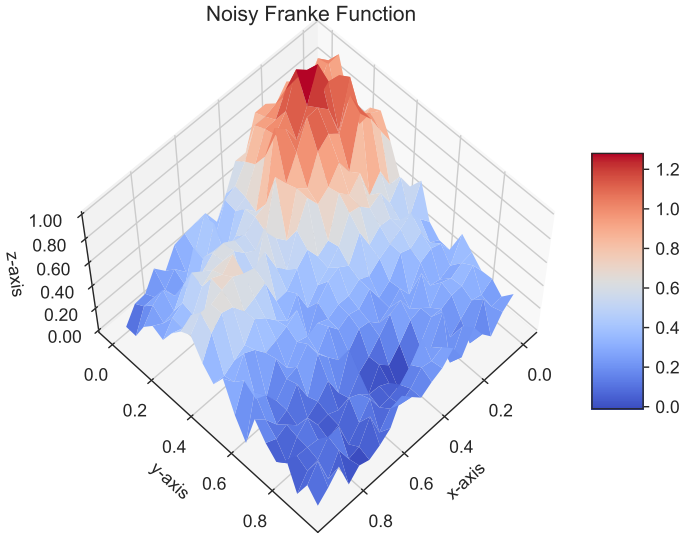


Figure 2: Franke function, defined in Equation 1, with added stochastic noise. The noise added in this figure had a standard deviation of 0.1.

2.1.2 Real Terrain

Real topographical data was fitted in the second part of this work. The data was defined for x and y [0,1], similar to the Franke function. Since the terrain data had much

finer features than the Franke data, 1000 steps were taken for each parameter inside the interval. The terrain data was obtained from https://earthexplorer.usgs.gov/, and is located near to Stavanger in Norway. The data is illustrated in Figure 3. Comparisons the Franke function and terrain data showed finer and more complex details in the terrain data.

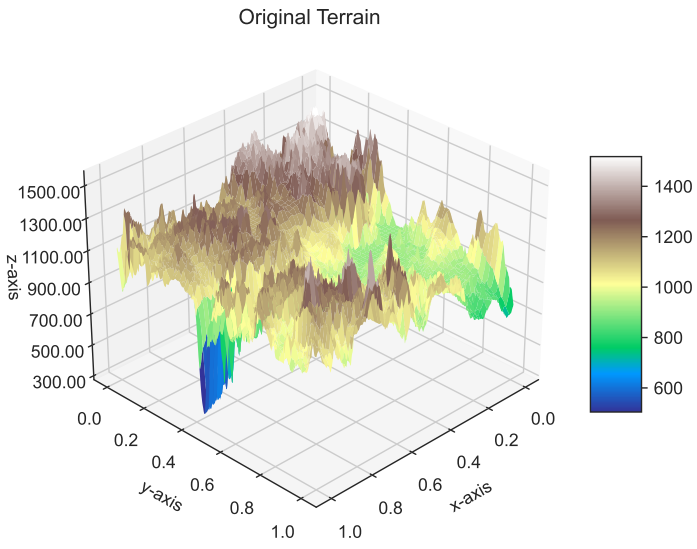


Figure 3: Terrain to be fitted

Machine learning algorithms generally need the input data to be scaled to the same scale [2]. However, for this work, the x and y data already had the same scale for both the Franke function and the terrain data, and was therefore not scaled.

2.2 Linear Regression Models

Regression is one of the major types of supervised machine learning algorithms, concerning the prediction of continuous numbers [3]. These kinds of algorithms are therefore suitable for predicting the elevation of both the Franke function and terrain data in each grid point.

Linear regression models make the predictions, or outputs, as a linear combination of the inputs [3, 7]. The output is the dependent variable y and the response from a set of independent variables x. One can assume this relationship to be the sum of a function $f(x)$ and some stochastic noise ϵ, which are assumed to be additive [8]. This relationship can be written as

$$y = f(x_1, ..., x_k) + \epsilon \tag{2}$$

where k is the number of input variables [8]. In linear regression, we want to estimate the systematic component $f(x)$ as a linear combination of the inputs. This gives us

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \epsilon \tag{3}$$

where the β values are unknown parameters [8]. The values predicted by the model can then be rewritten in matrix form as a vector $\tilde{\boldsymbol{y}}$ (length n):

$$\tilde{\boldsymbol{y}} = \mathbf{X}\boldsymbol{\beta} \tag{4}$$

where $\boldsymbol{X}$ is the design matrix (size n $\times$ p) which contains the x-values, and is known. $\boldsymbol{\beta}$ is a vector (length p) which contains the optimal parameters for the fit, and is unknown [8]. $\boldsymbol{\beta}$ is the problem the machine learning model will learn to predict when we fit data using linear regression [3].

Different linear regression models solve this problem in different ways. The main differences between the models are how they use the input data to solve for $\boldsymbol{\beta}$ and how they control the complexity of the model [3]. Linear regression models are generally based on the minimization of a cost function. A cost function gives the difference between the predicted and the true value [7]. Different linear regression models are associated with different cost functions, and machine learning algorithms can be used to minimize these.

For the regression in this work, polynomial functions of different degrees of x and y were used to fit the Franke function and the topographical data, where the elevation was defined as z. OLS, Ridge, and Lasso were employed as linear regression models. These models will now be presented.

2.2.1 Ordinary Least Squares

OLS is a standard model for linear regression due to its mathematical simplicity and statistical properties [3, 8]. This model finds the optimal $\boldsymbol{\beta}$, called $\hat{\boldsymbol{\beta}}$, from the sum of the squared differences (the mean squared error) between the predicted values and the true values [3]. The cost function for OLS is defined as:

$$C^{OLS}(\boldsymbol{\beta}) = \frac{1}{n}\left\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\} \tag{5}$$

OLS makes predictions by minimizing this cost function, which can be done by solving for the derivative of the function with respect to $\boldsymbol{\beta}$ being equal to zero [9]. Ignoring n, we get

$$\frac{\partial C^{OLS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T} = 0 = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{6}$$

The solution can be rewritten as:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{7}$$

where $\boldsymbol{X}$ is the design matrix containing the parameters of the n-th order polynomial, $\boldsymbol{X}^T$ is the transverse of the design matrix, $\boldsymbol{y}$ a vector with the real values [8, 9]. Additional derivations of expected values and variance of y_i and β are in Appendix C.

OLS is easy to implement, and is good at training data, but can easily become overfitted as it has no hyperparameters to control the complexity of the model [3]. Additionally, OLS requires $\boldsymbol{X}^T\boldsymbol{X}$ to be invertible, which is not always

the case. If $\boldsymbol{X}$ is singular, meaning rank-deficient due to linearly dependent columns, $\boldsymbol{X}^T\boldsymbol{X}$ is non-invertible, and there is no numerically stable solution for $\hat{\boldsymbol{\beta}}$ using OLS [8]. This problem can be solved using shrinkage methods, like Ridge and Lasso [9].

2.2.2 Ridge

Ridge is a linear regression model where the diagonal of the design matrix is adjusted with a variable λ, which is called the penalty. This was originally done in order to make the $\boldsymbol{X}^T\boldsymbol{X}$ invertible, by adjusting it with a small value [10]. However, λ is a hyperparameter which can also be optimized in order to improve the model, and Ridge can in cases perform better than OLS for small λ. Consequently, Ridge allows for control over the model complexity, which reduces the overfitting. Therefore, models obtained using Ridge are often more generalized than those obtained using OLS [3].

The cost function for Ridge regression is given by:

$$C^{Ridge}(\boldsymbol{X},\boldsymbol{\beta}) = \left\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\right\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \quad (8)$$

Similar as for OLS, the optimal parameters are found by solving for the derivative of the cost function with respect to $\boldsymbol{\beta}$ being equal to zero. The solution is found as

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \quad (9)$$

where $\boldsymbol{I}$ is the identity matrix, a diagonal matrix with values of 1 on the diagonal and 0 everywhere else. Note that Equation 9 differs from Equation 7 only by the $\lambda\boldsymbol{I}$ term. When λ becomes large, it drives the fitting coefficients to zero, while $\lambda = 0$ gives the same expression as OLS. If $\boldsymbol{X}^T\boldsymbol{X}$ is rank-deficient, $\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}$ will still be invertible for large λ [8].

It is common to exclude β_0, i.e., the intercept, from the penalization [8]. This is done automatically by `sklearn`.

2.2.3 Lasso

The final linear regression model which was investigated for the fitting was Lasso, which is an abbreviation of *least absolute shrinkage and selection operator*. Similar to Ridge, Lasso introduces a regularization term which penalizes the fitting coefficients. In contrast to Ridge, Lasso drives some coefficients all the way to zero, meaning they are excluded entirely from the fit, as a feature selection [3]. Lasso can therefore be useful when you have a large amount of unnecessary features, and can make it easier to interpret the model [8].

Lasso can also solve the problem of $\boldsymbol{X}^T\boldsymbol{X}$ not being invertible, and is also generally considered better at test data than OLS due to the control over model complexity introduced by the regularization term.

The cost function for Lasso regression is given by

$$C^{Lasso}(\boldsymbol{X},\boldsymbol{\beta}) = \left\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\right\} + \lambda||\boldsymbol{\beta}||_1 \quad (10)$$

where $||\boldsymbol{\beta}||_1 = \sum_i |\beta_i|$ denotes the norm-1 [9]. As before, we minimize the cost function by finding its derivative with respect to $\boldsymbol{\beta}$ and setting it equal to zero. Using Lasso, the optimal fitting parameters are found by:

$$\hat{\boldsymbol{\beta}}_i^{\text{Lasso}} = \begin{cases} y_i - \frac{\lambda}{2} & \text{if} \quad y_i > \frac{\lambda}{2} \\ y_i + \frac{\lambda}{2} & \text{if} \quad y_i < -\frac{\lambda}{2} \\ 0 & \text{if} \quad |y_i| \leq \frac{\lambda}{2} \end{cases} \quad (11)$$

Similar as for Ridge, one commonly excludes the intercept from the penalization, which is done automatically by `sklearn`.

Polynomials with different degrees were fitted to both the Franke function and the terrain data using all three linear regression models. For Ridge and Lasso, λ was also optimized.

2.3 Resampling Methods

Resampling methods are techniques to split and reuse data. These are common in machine learning due to the dependence on large data sets to obtain good models. Resampling is important to prevent overfitting of the models, which is when the model becomes too trained on the data set it has been trained on, and is not generalized and able to predict new data [3]. Overfitting is avoided using models with low complexity. On the other hand, using too simple models can result in underfitting. As a result, a compromise in complexity of the model must be made. This will be elaborated on in Section 2.4.1.

Resampling also allows for the use of statistical tools when examining data. Bootstrap and cross-validation were used as resampling methods in this work, and the performance of the methods were compared.

2.3.1 Bootstrap

Bootstrap is a resampling technique where you estimate a parameter of the population with n samples by drawing n numbers from the distribution with replacement [3]. This means that you do not remove the value you have measured from the population when you have measured it. Then, you evaluate the $\hat{\beta}$ based on the observations from your chosen samples. You repeat this process k times, and can then obtain a histogram of the obtained $\hat{\beta}$. This can be then treated as a probability distribution. Bootstrapping has advantages as not needing large sample sizes, and not relying on any assumption about the distribution of the data.

2.3.2 Cross-Validation

Cross-validation is another resampling technique, where the data is repeatably split into random training and test sets and evaluated [3, 8]. The model trains on the training set, while the test set is hidden, to be used for evaluation of the trained model. One typically divides around 80/20 into test and training data sets when splitting data [2]. When the model is trained using the training set, the performance is determined by its predictions of the values in the test

set. The reasons for splitting the data avoiding overfitting and to be able to estimate the generalization error of the model using the test set [2].

Cross-validation is more robust than splitting the data into a training and testing set once, where you could be lucky or unlucky with the splitting of the data, and obtained unrealistically high or low performance, respectively [3]. The drawback of cross-validation is the high computational cost [3].

Using a k-fold cross-validation, the data is split into k folds, and the model is trained and tested k times, each time with a new fold used as a test set and the other folds used as training sets [2]. In this work, cross-validation with 5 to 10 folds were tested.

The splitting into training and testing sets is important to assess the performance of the model before it is tested on measurement data [3]. Methods to evaluate the performance of the models will now be presented.

2.4 Model Evaluation

Evaluating model performance is an important step in model development, and the preferred choice of evaluation metrics depends on the type of model. Model evaluation for regression often employs the MSE, which accounts for bias and variance in a model, and the R^2, which is a measure of how well the model predicts the original data [3]. These two metrics will now be introduced.

2.4.1 Mean Squared Error

It is important to consider the bias-variance trade-off utilizing MSE for model evaluation. The bias of a model is determined by its ability of the model to approximate the training data, and is thus typically decreased by increasing model complexity. However, a too high model complexity can result in an increase of the variance of the model as it becomes overfitted and less generalized. This results in worse predictions of the test data. Consequently, there is a trade-off between the bias and variance of the model [2].

The bias-variance trade-off can be demonstrated by deriving the MSE in terms of bias and variance. The MSE is a cost function, and evaluates the difference between the predicted result and the real value. The standard MSE equation as defined by James et al. [11]:

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{12}$$

where y is the real value, $\tilde{y}$ is the approximated value provided by the model, and

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{13}$$

is the mean of y. Finding the expectation value of Equation 12 and expanding the expression, we obtain:

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

We will now examine each of the three terms separately. Starting with $\mathbb{E}\left[\boldsymbol{y}^2\right]$, we use that $y = f(x) + \epsilon$, and expand the term:

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

$$\mathbb{E}\left[\boldsymbol{y}^2\right] = \mathbb{E}\left[f^2\right] + \boldsymbol{\sigma}^2$$

For the last equality, we used the mean value and variance of ϵ, which follows the distribution $N \sim (0, \sigma^2)$. Examining the next term, we can again expand and use the fact that the mean value of ϵ is zero:

$$\mathbb{E}\left[\boldsymbol{y}\tilde{\boldsymbol{y}}\right] = \mathbb{E}\left[(f + \boldsymbol{\epsilon})\tilde{\boldsymbol{y}}\right] = \mathbb{E}\left[f\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\boldsymbol{\epsilon}\tilde{\boldsymbol{y}}\right] = f\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]$$

The last term is defined by Wieringen [12] as:

$$\mathbb{E}\left[\tilde{\boldsymbol{y}}^2\right] = var[\tilde{\boldsymbol{y}}] + (\mathbb{E}[\tilde{\boldsymbol{y}}])^2$$

Collecting and rearranging the terms, we can define MSE in terms of f and $\tilde{\boldsymbol{y}}$:

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \mathbb{E}[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \mathbb{E}[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \sigma^2$$

$$MSE(f, \tilde{\boldsymbol{y}}) = Bias[\tilde{\boldsymbol{y}}] + var[\tilde{\boldsymbol{y}}] + \sigma^2 \tag{14}$$

f is the output data from the model, which is non-stochastic, has no assumed distribution, and is assumed to be independent of ϵ. It is necessary to define MSE in terms of f instead of the $y = f + \epsilon$, since the bias term would be zero using y. Here it is seen that the MSE is the sum of the bias and the variance of the model, and the variance of the data itself [2].

2.4.2 R2 Score

The second evaluation metric used in this work is the R^2 score. This parameter evaluates how well the model predicts the output data, and is defined as:

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{15}$$

An R^2 of 1 implies a perfect model, while values close to 0 imply a poor model. R^2 can in general not be used to compare different models [8]. Consequently, MSE will be used as the main metric for model evaluation in this work. R^2 will be used along with MSE to evaluate each of the models to optimize the polynomial degree and hyperparameters.

The theory and method for the development of the models in this work have now been presented. The following section presents the results obtained using these models.

3 Results and Discussion

The results obtained in this work are presented in two parts. The first part concerns the fitting of the Franke function presented in Figure 1, while the second part contains the fitting of the topographical data presented in Figure 3. The methods presented in the previous section were used for the model development for both data sets.

As a starting point, the results obtained from the code written in this work were compared with those obtained using `sklearn`. The purpose of this comparison was to verify the performance of the code. The comparison was performed by comparing the MSE obtained for both OLS, Ridge, and Lasso regression of the Franke function, using a 10-fold cross-validation, fifth-order polynomials, with various values of λ. The resulting MSEs obtained using our own code and `sklearn` to perform the fits are compared in Figure 4. The left plot shows the obtained MSE values, while the right plot shows the difference in MSE for OLS and Ridge. As OLS has no λ, it has a constant MSE as a function of λ. The MSE is higher for Lasso than for OLS and Ridge. For Ridge and Lasso, the MSE increases with increasing λ above around 1E-4.

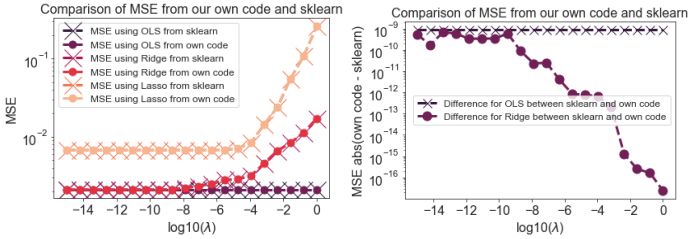


Figure 4: Difference between the MSE of our own code and the `sklearn cross_val_score` function. Our own data are presented by O-s and the `sklearn` data is presented by x-s.

From the left figure, it looks like the MSEs obtained using our own code are essentially identical to those obtained using `sklearn`, as the lines are seemingly overlapping. From the right figure, one can see that there is a finite difference between the obtained MSEs. However, this difference is of the order of 1E-9 or smaller, and will therefore not influence the values discussed in this work.

As the performance of the code written for this project has been verified, the results from the fitting of the Franke function will be presented.

3.1 Model Optimization to the Franke Function

This section presents the model optimization of the Franke function. For the purpose of obtaining more generalized models whic can make better predictions on the test data, resampling techniques were employed and optimized [3]. Then, the effect of different polynomial degrees and λ were explored. Finally, the models were compared, and the optimal fit was selected.

3.1.1 Resampling Methods

In this section, the performance of the resampling techniques bootstrap and cross-validation will be compared. Firstly, performance of fits made using bootstrap will be presented.

As seen from Equation 14, bias and variance are the two sources of error making up the MSE which is dependent on the model. To evaluate and minimize these, the bias, variance, and MSE as a function of model complexity were found for the Franke function fitted using an OLS model with 50 bootstrap resampling and 100x100 data points. The results are presented in Figure 5. Code to plot the bias-variance trade-off can be found on the Github repository.

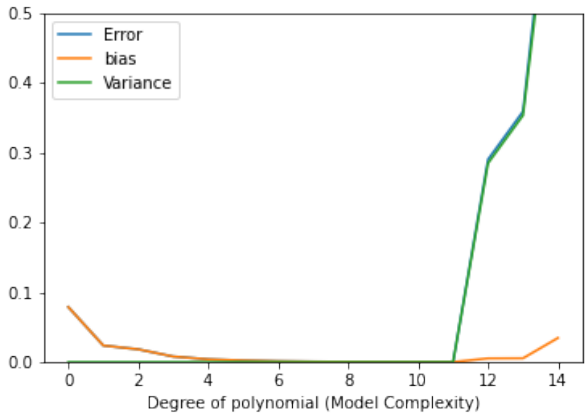


Figure 5: Bias-variance trade-off of OLS model of Franke function data with 50 bootstrap resampling and 100x100 data points.

In the figure, the model error and bias decrease with increasing model complexity. The variance remains fixed near zero until model complexity increases to the 11th-degree polynomial. At this point the bias-variance trade-off occurs, so that model error and variance increase with model complexity and bias remains low. While the bias-variance trade-off occurs at the 11th-degree polynomial there was no significant improvement in model error since the fifth-degree polynomial. This could indicate that the fifth-degree polynomial is the best balance of model complexity and error.

Figure 6 presents the MSE obtained from the testing and training data as a function of polynomial degree. The training and testing error consistently decreased with model complexity until the model error was dominated by the variance at the 11th polynomial degree. It is important to note that a model with zero training error is overfit to the training data and will typically generalize poorly [10]. This occurs as the model approaches the seventh polynomial degree in Figure 6.

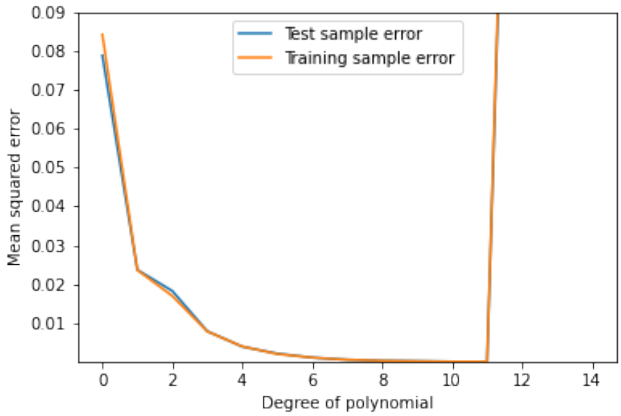


Figure 6: Training and test error for OLS model of the Franke function.

The performance of model fits performed using bootstrap and cross-validation was compared in order to select the optimal resampling method for this work. This comparison is provided in Table 1. Code providing the cross-validation MSE values is found at Github.

Table 1: Comparison of the MSE obtained using bootstrap with 50 resamplings and cross-validation with 10 folds.

Polynomial degree	Bootstrapping MSE	Cross-validation MSE
1	0.024	0.021
2	0.0183	0.0175
3	0.0079	0.0074
4	0.0040	0.0025
5	0.0023	0.0021

As seen in Table 1, cross-validation outperformed bootstrap for all the polynomial degrees. This could be because the bootstrap is training on the same data set without changing it, in contrast to cross-validation. It was discussed in the theory section that not changing the training and testing sets could induce unrealistic model evaluation scores depending on how lucky one is with the selection of the training and test sets [3].

Cross-validation was used as a resampling technique for the rest of the fitting of the Franke function, both because of its higher performance and because of its increased reliability compared to bootstrap.

3.1.2 K-fold cross-validation

The number of folds for the k-fold cross-validation must be determined. Upon increasing the number of folds, there is a trade-off between a potentially increased performance of the fit, and a higher computational power required. It is common to use between 5 and 10 folds, and this range was therefore used as a starting point for the evaluation.

The MSE for OLS, Ridge, and Lasso for different numbers of folds is shown in Figure 7. The fifth-order polynomials and λ of 1E-10 were used for the comparison, based on preliminary experiments. The MSE of OLS and Ridge was similar while the MSE of Lasso was higher for all the numbers of folds.

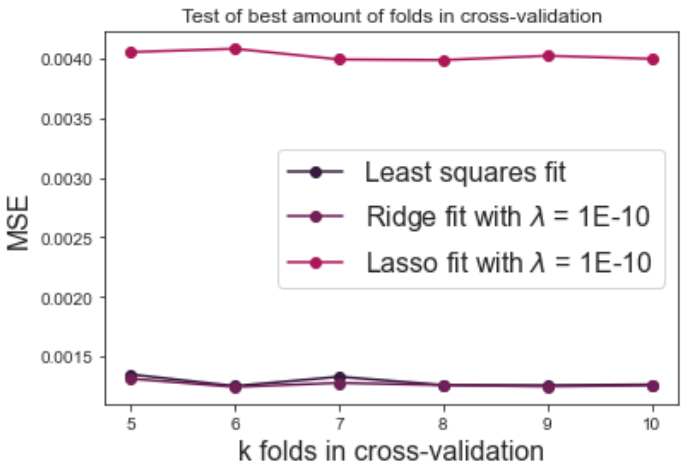


Figure 7: MSE of simulations of the Franke function using different simulation models and a different number of folds in the cross-validation.

The MSE values in Figure 7 were not significantly influenced by the number of folds within the chosen range for any of the regression models. A number of 5 to 7 folds gave slightly higher MSE than higher numbers of folds, while a number of 9 folds gave a higher MSE for Lasso than what 8 and 9 did. 10 folds gave a slightly lower MSE than the lower value folds for all three models. Additionally, a number of 10 folds is commonly used in the field. Consequently, 10-fold cross-validation will be used for the remainder of the evaluation of the Franke function. It is however important to note that the increase in folds is associated with a higher computational cost [3], and a lower number of folds could be considered for more computationally heavy data sets.

3.1.3 Polynomial degree

Upon having optimized the resampling method with the number of folds, the polynomial degree of the model was optimized. This was done by comparing the accuracy of the fits of polynomials with different degrees while using a 10-fold cross-validation.

The degrees of x and y were varied individually. Initial tests revealed a minimal variation in MSE beyond the fifth degree of both x and y for all models when approximating the Franke function. Therefore, these high-degree polynomials were not further considered due to their increased computational costs compared to lower-order polynomials. Consequently, models with polynomial degrees from the first to the fifth were evaluated, as shown in Figure 8. All the curves follow and S-shape, with decreasing MSE and

increasing R^2 with increasing polynomial degrees of both x and y.

Note that the R^2 score should not be used to compare the performance of the different models, only the different polynomial degrees [8].

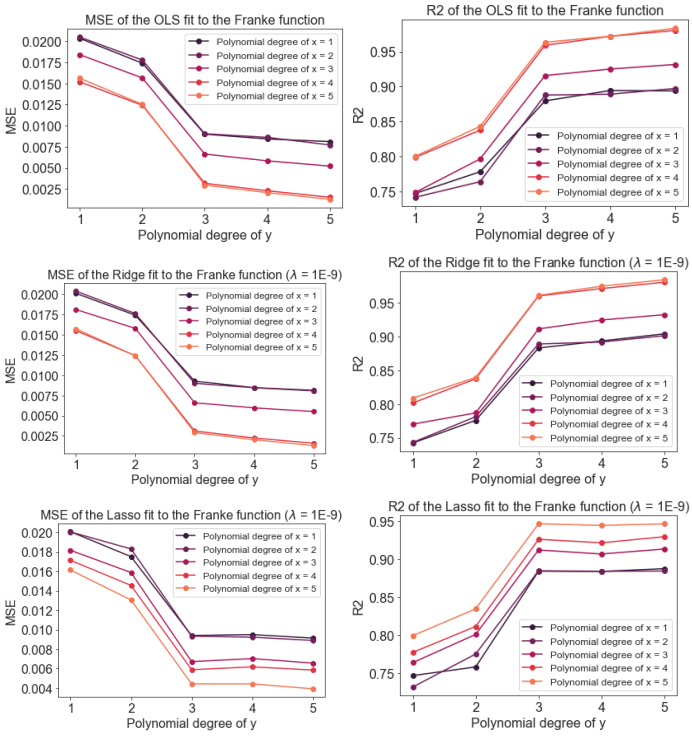


Figure 8: MSE and R^2 evaluation of simulations of the Franke function for different polynomial degrees. The upper two plots show OLS, the middle plots show Ridge, and the bottom plots show Lasso.

Comparing the MSE of the different models, it is clear that OLS and Ridge provide very similar results. Lasso, despite having similar MSE values as the other models for the lower-degree polynomials, has significantly higher MSE values for higher-degree polynomials. This could be explained by the fact that the Franke function is a polynomial function, thus it could be expected that a polynomial function will produce the best fit. With a low value for λ, the Ridge simulation is very similar to that of the OLS. When we introduce more complexity to our model, as in the Lasso-regression, the chances of producing errors are greater. Lasso has an advantage for models with many unnecessary features as it can press some of the estimators to zero [8]. However, that was not the case for the Franke function, which is a simple function. Consequently, OLS and Ridge provided the better fits.

The degrees of x and y were varied individually for the different fits, resulting in slightly different MSE and R^2 for all three models. This was particularly noticeable for the low degrees. To illustrate, the MSE for the OLS and Ridge fit for a constant first-degree y was almost identical for both first- and second-degree x, with the data points lying essentially on top of each other. Moving to a constant second-degree y, the MSE was still almost identical for first- and second-degree x. However, there was a significant difference between the MSE of first- and second-degree y for constant first- or second-degree x. For the high degrees for OLS and Ridge, the MSE is very similar for fourth- and fifth-degree of x, and for third-, fourth- and fifth-degree of y. This difference between the performance in x and y is a result of the Franke function not being symmetrical in the x and y directions.

The performance of OLS and Ridge was high for the fourth and fifth degrees of both x and y, with very little difference between the two degrees. On the other hand, the MSE from the Lasso-regression was still decreasing as a function of the polynomial degree even after this point. This could be due to the fact that the OLS and Ridge fit managed to produce a very low MSE value for Franke function fit for these relatively low polynomial degrees, making it difficult to obtain further improvement from the already quite good models. In contrast, for the Lasso-regression, the fit was not as good, thus it could be improved to a larger extent by increasing the polynomial degree.

To conclude, a fourth-order polynomial of both x and y provided a good fit for both the OLS and Ridge models. As there is a computational cost of increasing the polynomial degree and there was very little improvement in MSE by doing so, this polynomial degree would be considered optimal for fitting the Franke function with these given regression models. However, for the Lasso model, there was a significant improvement of MSE upon increasing from the fourth and to the fifth degree, particularly for the degree of x. Based on the small size of the data set, the computational cost of fitting a fifth-degree polynomial compared to a fourth-degree one is small. Consequently, in order to compare the three models during the rest of this work, fifth-degree polynomials will be used for all the models.

3.1.4 Beta-values as a Function of Polynomial Degree

The β values for the OLS, Ridge, and Lasso models were introduced in Equations 7, 9, and 11, respectively. We will now study how these values are influenced by the polynomial degree used for the fit of the models. For this study, a small and randomly selected portion of β values from the training sets is presented as a function of the polynomial degree in Figure 9. A λ of 1E-9 was used.

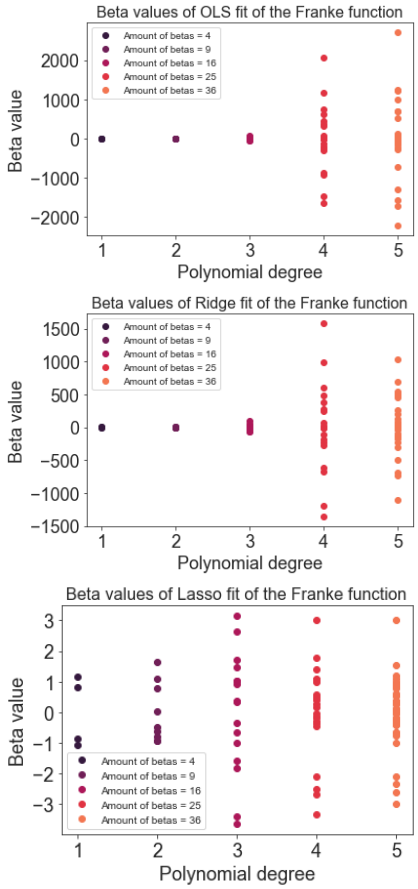


Figure 9: The distribution of randomly selected β values from the training sets as a function of polynomial degree for fits of the Franke function. λ = 1E-9.

The β values correspond to the change in a dependent variable as a response to a change in the independent variables for the fits. In this context, the values of β will represent how much the z-value of the Franke function changes when the x and y values change. For OLS and Ridge, the β values spread out for an increasing polynomial degree. This indicates that polynomials with higher degrees resulted in a higher fluctuation of the z-values than lower degrees. For Lasso, the spread of the values was more constant, not as influenced by the polynomial degree.

The different models had different values of β, where the absolute values were much larger for the OLS than the Ridge, and the β-values for the Lasso-regression were several orders of magnitude lower than the two others. The reason for the higher values for the β of OLS compared to the two other regression models is the penalization of the β introduced by λ for Ridge and Lasso. This suppresses the β, as was seen in Equation 9 and 11. The values for β for Ridge were less suppressed than those of Lasso. λ was only 1E-9 for both models, and for Ridge, this only corresponds to a small decrease in the values of β relative to that of OLS. For Lasso, however, some of the β were pressed to zero.

3.1.5 Choice of λ

So far, a 10-fold cross-validation and fifth-order polynomials have been decided for the fit of the Franke function, based on the trade-off between model performance and computational cost. The last parameter to optimize is the λ for the Ridge and Lasso regression models. Again, a preliminary screening was performed, revealing values of λ above 1E-5 to give high errors and values below 1E-18 to result in very small changes in MSE. Consequently, λ between 1E-18 and 1E-5 were compared for both models. The reuslts are presented in Figure 10.

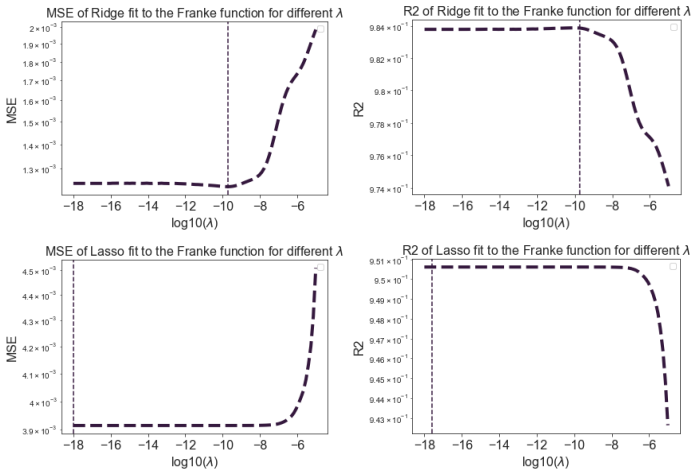


Figure 10: MSE and R^2 as a function of λ for Ridge and Lasso fits of the Franke function. A polynomial degree of 5 was used for all simulations.

Comparing the MSE of Ridge to that of Lasso, the former outperformed the latter for all the λ values that were evaluated. The reason for this has already been elaborated on, and explained by the simplicity of the Franke function. The λ values resulting in the lowest MSE and highest R^2 values for each of the models were marked with vertical grey lines. For Ridge, this value was around 1E-9, while for Lasso it was around 1E-18. The performance of Lasso was steady up to around 1E-7, while the performance of Ridge decreased already above 1E-9. However, for Ridge, there was a small dip in the MSE around 1E-10 to 1E-9.

The low optimal λ values and the general lack of improvement compared to OLS indicate that the biasing of features that were introduced in the Ridge and Lasso methods was not needed for the fitting of the Franke function. Particularly high penalty factors made the models significantly worse.

3.1.6 Influence of Noise on Fit

Stochastic noise was added to the Franke function to study its influence on the fit. Figure 2 illustrates the Franke function with an added noise of 0.1, meaning the noise had a standard deviation of 0.1 in the z-direction. For higher degrees of noise, the surface became less and less coherent, with more irregularities and spikes. It could be interesting

to compare the noisy Franke function to the illustration of the terrain in Figure 3 which has even more spikes and irregularities.

The influence of the fits of the different models on increasing noise is illustrated in Figure 11. The zero added noise represents the original Franke function. Since the added noise was random, giving a different function each run, the noise results were averaged over 1000 runs, to create a reproducible set of data.

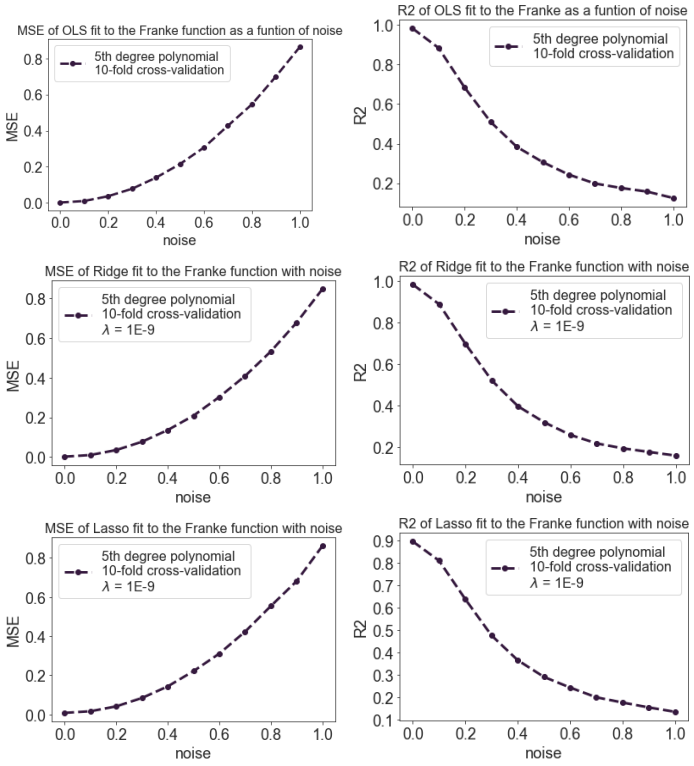


Figure 11: MSE and R^2 of the fitting of the Franke function for fifth-degree polynomials and 10-fold cross-validation for all three models, as functions of stochastic noise. The noise has a standard deviation ranging from 0 to 1, and the MSE and R^2 values the mean values of a total of 1000 simulations.

An increasing noise resulted in increasing errors for all the models. This is because the noise had a standard deviation, from 0 to 1 in this plot, but not a displaced mean value. This makes the predictions for the models worse for the increasing noise. This can be seen from Equation 14, where MSE is written as a function of the bias, variance, and the variance of the data, σ^2. The σ is the standard deviation of the noise. Consequently, increasing noise directly resulted in an increased MSE. MSE increased to 1 since the noise had a standard deviation of 1.

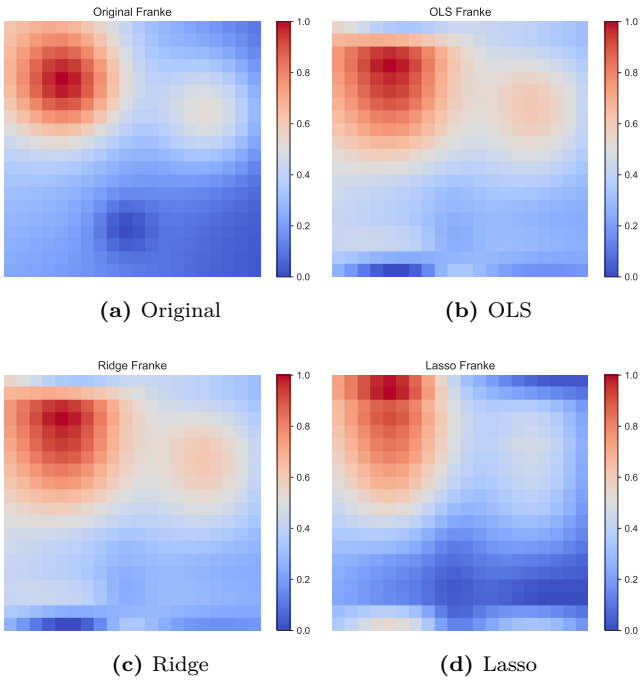
The effect of noise was similar for all three models. The MSE increased seemingly exponentially from a low value up to around 1 for all models when the noise increased to 1. Correspondingly, the R^2 value decreased towards 0. The reason for the similar trend in all models can be seen from Equation 14 again. The added σ^2 will be the same for all the models as it is a property of the data set itself, irregardless of the bias and variance of the model.

3.1.7 Comparing the results for best fit

The regression models have now been optimized in terms of resampling method, polynomial degree, and λ. The MSE values for the optimized OLS, Ridge, and Lasso models for the Franke function are tabulated in Table 2. The MSE/Mean value denotes the MSE normalized for the given data. Figure 12 shows the Franke function illustrated in 2D, along with the obtained optimized fits. The difference between the values obtained by the fit and the real values for the Franke function are shown in 3D in Appendix A, along with the difference between the fits obtained by OLS and Ridge, illustrating their similarity, but that they are not identical.

Table 2: Table showing the MSE and MSE/Mean value of the Franke function, from a 10-fold cross-validation, for the fit using OSL, Ridge and Lasso models with optimized parameters. A polynomial degree of 5 is chosen for all simulations. For Ridge, λ = 1E-9 and for Lasso, λ = 1E-18.

Model	MSE	MSE/Mean data value
OSL	0.00130	0.00306
Ridge (λ = 1E-9)	0.00128	0.00300
Lasso (λ = 1E-18)	0.00836	0.01965



(a) Original **(b)** OLS

(c) Ridge **(d)** Lasso

Figure 12: 2D plots of the original Franke function, along with the fits obtained using OLS, Ridge, and Lasso. The fits were obtained using a 5th order polynomial and 10-fold cross-validation. For Ridge and Lasso, λ was 10^{-10}.

From both Table 2 and Figure 12 it is clear that Lasso performs poorly compared to Ridge and OLS. We know that Ridge performs better when most predictors impact the response, while Lasso performs better in cases with few predictors impacting the response. For the Franke function, the former seems to be the case, explaining why the Ridge model performs the best out of the two.

OLS generally performed better than Ridge. However, the performance of Ridge increased for lower values of λ, performing better than OLS for the value of 1E-9. Judging from the data in Table 2, the OLS and Ridge models were almost equally good at predicting the Franke function under the given conditions. It is possible that increasing the polynomial degree could change this, but given that satisfactory low MSE values already obtained, it was decided not to due to the simplicity of the data set. Based on the MSE values in Table 2, Ridge provided a slightly better fit than OLS. However, computational efficiency of performing the fits should also be considered. OLS is the most efficient out of the two models, as it requires fewer flops to perform an analysis than Ridge. Therefore, OLS was chosen as the optimal model for the fit of the Franke function.

The OLS model which was concluded to give the best fit is illustrated in Figure 19. Compared to Figure 1 illustrating the original Franke function, one can see that the model gives a good fit, but not a perfect fit. This can also be seen comparing the 2D figures in Figure 12. The optimal fits for OLS and Lasso were also illustrated in 3D, found at Github.

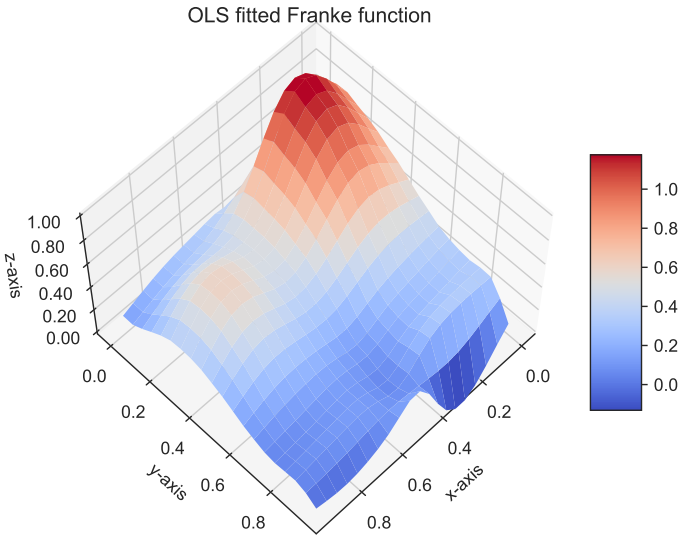


Figure 13: 3D representation of the best fit obtained for the Franke function. The fit was performed using OLS regression with a 10-fold cross-validation, and a polynomial degree of 5.

In conclusion, the linear regression models managed to fit the Franke function with quite good precision, based on the model evaluation metrics. The computational cost was low for OLS and Ridge, based on the running time of the functions, but higher for Lasso. The main features of the Franke function were captured by all the models, based on Figure 12. However, there were errors particularly around the highest and lowest values of y, as can be seen in Appendix A. This could be a result of overfitting.

3.2 Analysis of Terrain

The optimal model for fitting the Franke function has now been found. The second part of the work concerns fitting real topographical data. The same method for the development of the model as was used for the Franke function was used for this second part of this work. The data chosen for this study is provided in `Terrain1.tiff` and is from a terrain in Norway, close to Stavanger. This terrain was represented in 3D in Figure 3.

Note that the z-values in the terrain data go from around 300 to 1500. For the Franke function, the z-values ranged from around 0 to a little above 1. As the MSE, defined in Equation 12 is given by the difference in the z-value between the real and the approximated value, the MSE values obtained for the fitting of the terrain data can not be compared directly to those obtained for the fitting of the Franke function, due to the different scales of the data. However, the MSE/Mean value and trends can still be compared.

3.2.1 K-fold

For the Franke function, cross-validation was chosen as a resampling technique over bootstrapping, using 10 folds. Due to the increased complexity of the terrain data over the Franke function, the same resampling method with the same number of folds were chosen for this data. An increased number of folds was not chosen due to the increased computational cost.

3.2.2 Polynomial degree

The optimal polynomial degree for the model was chosen based on the MSE and the computational cost. Due to the increased complexity of the terrain data, higher order polynomial degrees were considered than for the Franke function. The polynomial degree of x and y were consequently not considered individually, to simplify the discussion. Figure 14 shows the MSE for OLS, Ridge, and Lasso for increasing polynomial degrees, where the degree denotes the degree of both x and y.

The MSE of OLS and Ridge was lower than that of Lasso for polynomial degrees up to and including the fifth. After that, the MSE of Ridge and particularly OLS increased drastically. On the other hand, the MSE of Lasso was seemingly quite constant independent of polynomial degree. For both OLS and Ridge, the polynomial degree fifth gave the lowest MSE.

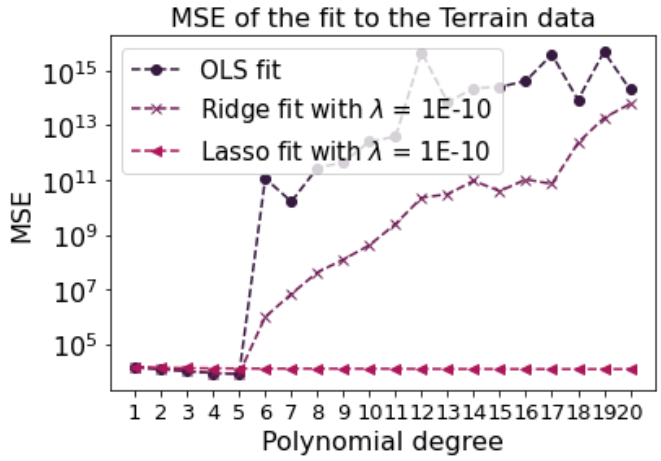


Figure 14: Plot showing how the MSE varies for the simulation of the terrain data as a function of polynomial degree (both x and y) for OLS, Ridge, and Lasso models.

In order to study the influence of polynomial degree on the MSE of Lasso more closely, Figure 15 shows the MSE of the Lasso fit. In this figure, it is easy to see that the MSE of Lasso regression decreased until the 14th-degree polynomial, before it started to increase.

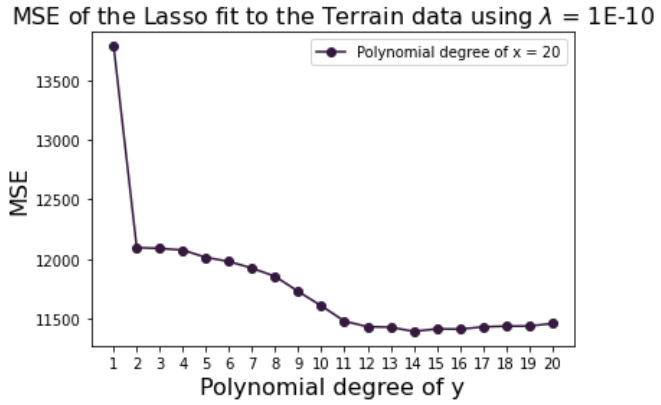


Figure 15: Plot showing how the MSE varies for the simulation of the terrain data as a function of polynomial degree for a Lasso model.

In conclusion, the fifth-degree polynomial gave the best fit for OLS and Ridge, while the 14th-degree polynomial gave the best fit for Lasso. Since the difference between the MSE of the fifth- and 14th-degree polynomial was significant for Lasso, we will proceed with these different polynomial degrees for the three models, in contrast to what was decided for the Franke function.

Compared with the Franke function, the optimal polynomial degree was the same for OLS and Ridge, but not for Lasso. Additionally, the values for the MSE were significantly higher for the terrain fit than for the Franke fit, which has been explained to be due to the higher z-values of the terrain data than of the Franke data. Furthermore, the terrain data had more "noise" which could further increase the MSE.

3.2.3 Beta values as a function of polynomial degree

As for the Franke function, we will study the distribution of the β values as a function of the polynomial degree of the fit. The β values were again randomly selected from the training set. The results are presented in Figure 16. To simplify, polynomial degrees up to the fifth were used for this study, despite the higher degree chosen for Lasso. Note that the scale for OLS and Ridge is given in 1E7, while the scale for Lasso is not. The λ was 1E-9 for Ridge and 1E-2.4 for Lasso based on model optimization which will be discussed in the following section.

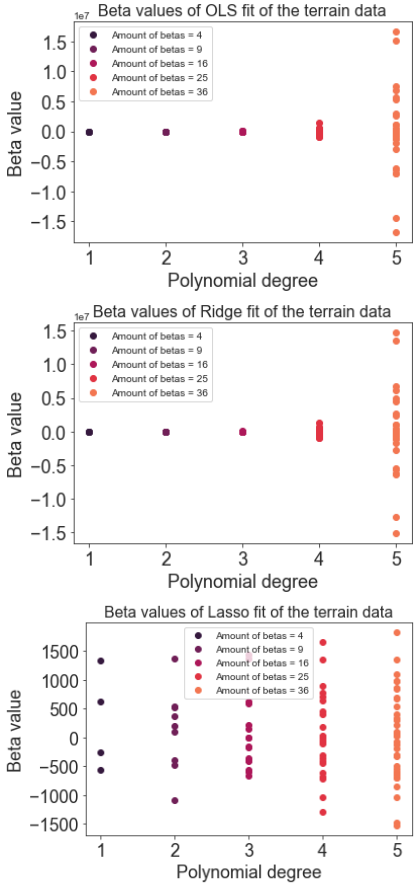


Figure 16: Plots showing the distribution of betas as a function of the polynomial degree of fits to the terrain data with $\lambda = 1E-9$ for the Ridge model, and $\lambda = 1E-2.4$ for the Lasso model.

As for the MSE values, the β values obtained here cannot be compared directly to those obtained for for Franke function due to the differences in the z-values. However, trends can still be compared.

The values of β increased as a function of polynomial degree for OLS and Ridge, as for the Franke function fit. Again, this can be explained by higher polynomial degrees giving stronger fluctuations of z than lower degrees. For

the Franke fit, the OLS and Ridge β values were similar, while for the terrain fit they seem close to identical, despite the same λ used for the two fits. The Lasso model seemed to have a rather stable distribution, even for an increasing number of β. This was also similar was observed for the Franke fit.

The β value for the OLS and Ridge fits were higher than for the Lasso fit, as for the Franke function. It is important to remember that different λ values were used for Ridge and Lasso, where the significantly higher value for Lasso suppressed the β values more than the lower λ values of Ridge did. The suppression of the β was formulated in Equations 9 and 11 for the two models. For OLS, there was no suppression.

3.2.4 Choice of lambda

The values of λ for Ridge and Lasso were optimized by studying its influence on the MSE for the terrain fit. As for the Franke fit, initial studies were performed to screen for a relevant range, and the results in Figure 17 are based on these initial screenings. In contrast with what was observed for the Franke fit, different ranges of λ were relevant for Ridge and Lasso for the terrain fit. Therefore, different values are shown in the figures for the two models. Cross-validation with 10 folds using polynomials with the fifth-degree for Ridge and 14th-degree for Lasso was used for the figures, as has been established as the optimal parameters so far. Vertical lines mark the optimal values of λ for each of the two models.

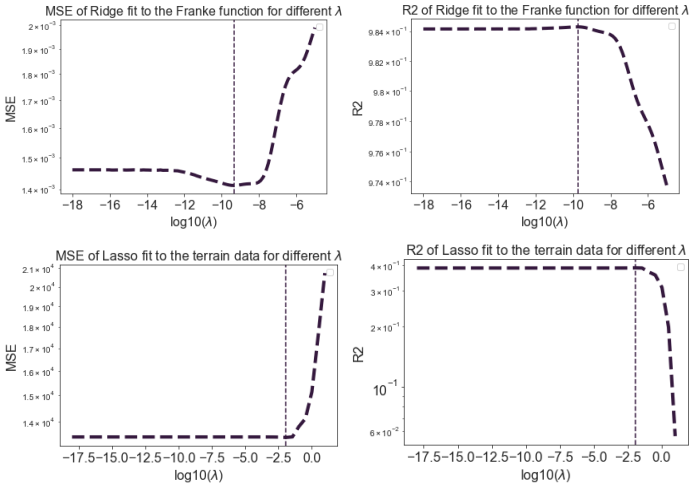


Figure 17: MSE and R^2 for the fit of the terrain data, using Ridge and Lasso, as a function of λ. A polynomial degree of 5 is used for the Ridge model, while a polynomial degree of 14 is used for the Lasso model.

According to the simulations illustrated in Figure 17, the optimal value for λ for the Ridge model was around 1E-9. This was the same value as was optimal for the fitting of the Franke function, and for the same polynomial degree.

For the Lasso model, the optimal value was $\lambda \approx$ 1E-2.4. This indicates that the biasing introduced by λ was more important for predicting the terrain data than it was for the Franke function. In addition to regularization of β, Lasso excludes certain features from the model completely, as a feature selection, which is useful when you have many unnecessary features [3, 8]. The high optimal value of λ for Lasso could imply that the model performs better when a larger number of features is excluded from the fit. The terrain data contained many fine features in contrast to the Franke function, which contained few and more distinguished features. This could explain why feature exclusion was more important for the fitting of the terrain than for the Franke function.

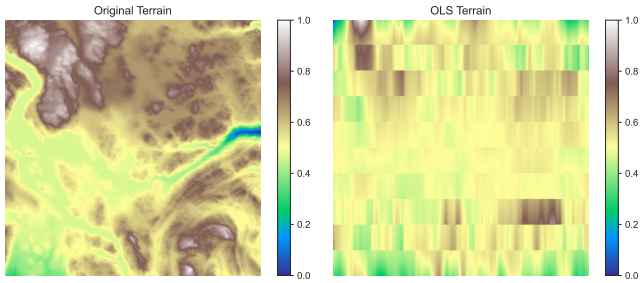
3.2.5 Optimal fit for the terrain data

The optimal fitting parameters for the three regression models have now been obtained for the terrain data by selecting the optimal resampling, polynomial degree, and λ. In this final section, the three optimal models will be compared based on their MSE and required computational power, as was done for the Franke function. The MSEs obtained for the three models are tabulated in Table 3. The λ values for Ridge and Lasso are presented in the table. The polynomial degree was five for OLS and Ridge, and 14 for Lasso, as has been concluded.

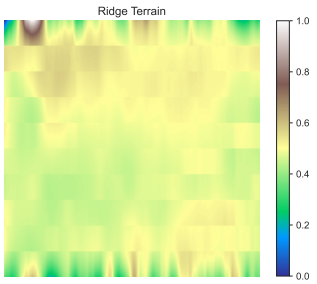
A 2D prepresentation of the original terrain is compared to the fits in Figure 18. Note that the fit for the Lasso regression is not included in this figure, as the plot was too computationally heavy to run even upon adjusting the tolerance and the maximum number of iterations. The reason for this is most likely the combination of the high-fold cross-validation and the high polynomial degree of the fit, combined with the larger size of the terrain data set compared to the Franke function. 3D representations of the difference between the original terrain data and the fits are provided in Appendix B.

Table 3: MSE and MSE/mean value of the terrain data, of the fit of our three different models with optimal parameters. A polynomial with degree five was used for the OLS and Ridge fit, and degree 14 for the Lasso fit.

Model	MSE	MSE/Mean data value
OLS	8138	7.4729
Ridge (λ = 1E-9)	8135	7.4702
Lasso (λ = 1E-2.4)	9411	8.6419



(a) Original **(b)** OLS



(c) Ridge

Figure 18: 2D plots of the original terrain data function, along with the fits obtained using OLS and Ridge. The fits were obtained using a fifth-order polynomial and 10-fold cross-validation. For Ridge, λ was 10^{-9}.

The results in Table 3 show the OLS and Ridge MSE to be significantly lower than Lasso for the terrain data. This is explained by Lasso being better at fitting data where few predictors are responsible for the response. On the other hand, Ridge regression is better at fitting data where many predictors affect the response. Looking at the 3D projection of the original terrain data from Figure 3, this data could be too chaotic to be represented by a few predictors, as is favored by a Lasso model. Additionally, the optimal Lasso model used a higher polynomial degree, and was much more computationally demanding, observed by it being much slower than the other models. Consequently, the Lasso model will not be chosen as the optimal model for the terrain fit.

The relative difference in MSE between the OLS and Ridge fit was very small, being well within any uncertainty one could expect. As for the fit of the Franke function, the Ridge model gave a slightly lower MSE than OLS. However, the relative difference in MSE between the two models was significantly lower for the terrain fit than for the Franke fit. Furthermore, the terrain data was more computationally heavy to fit. Therefore, we concluded that OLS provided the preferred fit for the terrain data.

The OLS fit for the terrain data providing the best fit is illustrated in 3D in Figure 19. A similar 3D plot for the optimal fit for the Ridge model can be found on Github, while the optimal fit for the Lasso model would not run, as for the 2D plot.

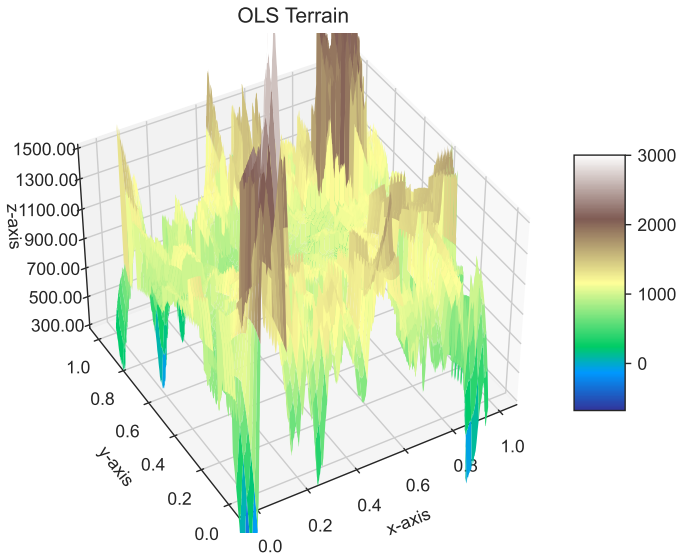


Figure 19: 3D representation of the best fit obtained for the terrain data. The fit was performed using OLS regression with a 10-fold cross-validation and a polynomial degree of 5.

Comparing the OLS fit in Figure 19 to the original terrain data in Figure 3, one can see that there are many large differences between the fit and the true data. The fit gave some peaks with very high values. The OLS did not manage to fit the terrain data very accurately.

Based on the figures in Appendix A and Appendix B, the fit for the Franke function was much more successful than that of the terrain data. Furthermore, this is seen from the MSE/Mean value for the two fits. Spatial correlation is important to obtain a good fit for simple regression models, which is higher for the smoother Franke function than for the terrain [1]. The fit of the Franke function had much fewer systematic deviations from the original data than the terrain. The differences between the original data and the fits were still largest around the smallest and largest y-values for the terrain, but the fit was no longer good in the center, as it was for Franke. Studying the difference between the OLS and Ridge fit in Appendix A and Appendix B, this was also much larger for the terrain fits than for the Franke fits.

4 Conclusion and Future Work

In conclusion, the OLS and Ridge approaches seemed to be favorable to the Lasso method, both in the case of the Franke function and the terrain data. This was assumed to be because the Lasso model works best in cases where only a few predictors impact the response, while the opposite is true for the Ridge model. Also, with a quite small value for λ, the Ridge model is similar to that of the OLS model, which is shown in the results as well.

It seemed that the OLS model was the optimal choice for both the Franke function and the terrain data. This conclusion was made as OLS produced approximately the

same error as the Ridge model, but the Ridge model was slightly more computationally demanding, as it contains some extra flops, making the OLS the most efficient method in both cases.

Cross-validation proved more successful than bootstrapping. This could be explained by the repeated reshuffling of the training and test data providing a more accurate model than only splitting and verifying the model once.

For future work, particularly the fit for the real terrain data could be improved. Suggestions for how to improve the fit could be to use other machine learning techniques. For example, neural networks have the potential to develop good models when dealing with complex data sets.

Additionally, the same approach used for this work could be tested on other terrain data for the purpose of exploring if there exist data sets where the Lasso model produced the optimal result.

Furthermore, it would be interesting to see if we could find a data set where the Ridge model and the OLS were more different than they have been in this project. This would mean a set of data where a higher value for λ would produce a lower MSE.

In this project, we have looked at a test function and geographical data. It would be interesting to apply this approach to topological data of a different kind, such as medicinal or economic data, to try and create models for real, societal features.

Overall, this has been a very interesting and fun project, and a great introduction to machine learning. This includes both basic coding examples, as well as being introduced to common machine learning concepts and algorithms.

References

[1] Maculotti, Giacomo, Gianfranco Genta, Danilo Quagliotti, Maurizio Galetto, and Hans N Hansen: *Gaussian process regression-based detection and correction of disturbances in surface topography measurements*. Quality and Reliability Engineering International, 38(3):1501–1518, 2022.

[2] Géron, Aurélien: *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[3] Müller, Andreas C and Sarah Guido: *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[4] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay: *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011.

[5] Hunter, J. D.: *Matplotlib: A 2d graphics environment*. Computing in Science & Engineering, 9(3):90–95, 2007.

[6] Haaland, Ben and Peter Z. G. Qian: *Accurate emulators for large-scale computer experiments*. 39(6), 2011. https://doi.org/10.1214%2F11-aos929.

[7] Murphy, Kevin P.: *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambdridge, Massachusetts, 2012.

[8] Fahrmeir, Ludwig, Thomas Kneib, Stefan Lang, and Brian Marx: *Regression models*. Springer, 2013.

[9] Hjorth-Jensen, Morten: *Fys-stk4155 data analysis and machine learning lecture notes*, 2023.

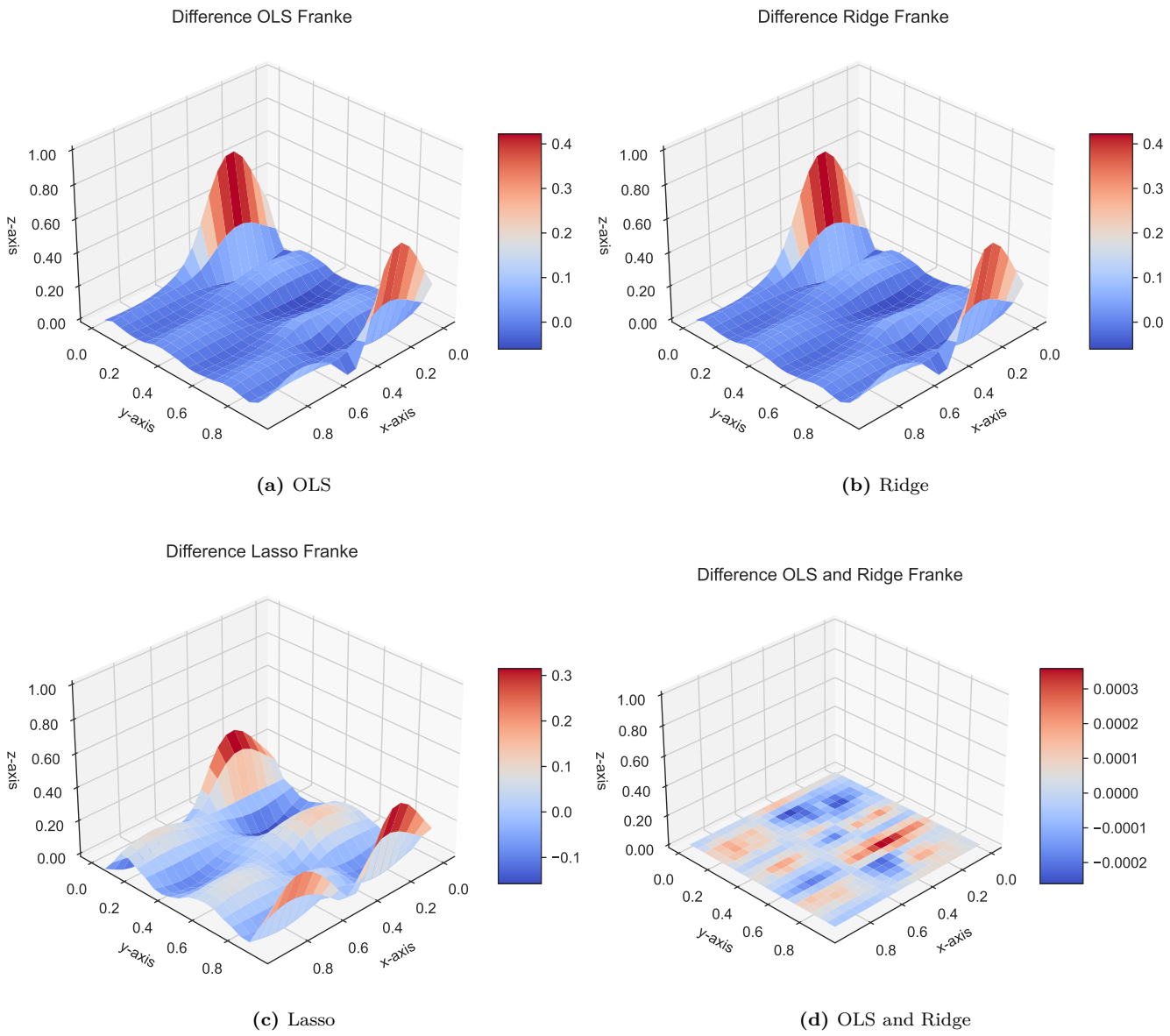
[10] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman: *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, Berlin, 2009.

[11] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani: *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. https://faculty.marshall.usc.edu/gareth-james/ISL/.

[12] Wieringen, Wessel N. van: *Lecture notes on ridge regression*, 2023.

Appendices

Appendix A Difference Data and Models: Franke Function



(a) OLS

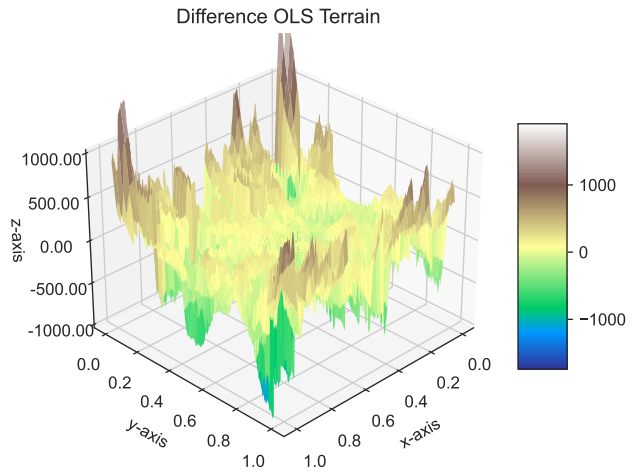
(b) Ridge

(c) Lasso

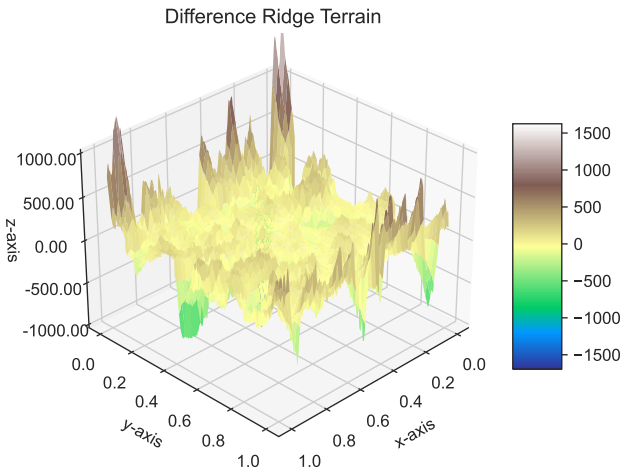
(d) OLS and Ridge

Figure A.20: 3D plots representing the difference between the actual values for the Franke function, and those predicted by the fits obtained using OLS, Ridge, and Lasso. The fits were obtained using a fifth-order polynomial and 10-fold cross-validation. For Ridge and Lasso, λ was 10^{-10}. The last figure illustrates the difference between the fits obtained using OLS and Ridge.

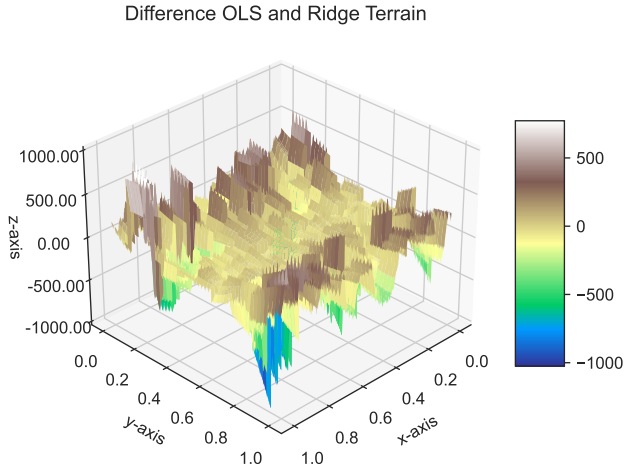
Appendix B Difference Data and Models: Terrain



(a) OLS



(b) Ridge



(c) OLS and Ridge

Figure B.21: 3D plots representing the difference between the actual values for the terrain data, and those predicted by the fits obtained using OLS and Ridge. The fits were obtained using a fifth-order polynomial and 10-fold cross-validation. For Ridge, λ was 10^{-9}. The last figure illustrates the difference between the fits obtained using OLS and Ridge.

Appendix C Derivations

The assumption we have made is that there exists a continuous function $f(\boldsymbol{x})$ and a normally distributed error $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ which describes our data data

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

We then approximate this function $f(\boldsymbol{x})$ with our model $\tilde{\boldsymbol{y}}$ from the solution of the linear regression equations (ordinary least squares OLS), that is our function f is approximated by $\tilde{\boldsymbol{y}}$ where we minimized $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$, with

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

The matrix $\boldsymbol{X}$ is the so-called design or feature matrix.

The expectation value of $\boldsymbol{y}$ can be shown for a given element i as [12]:

$$\mathbb{E}[(y_i)] = \mathbb{E}[(X_{i,*}\ \beta)] + \mathbb{E}[(\epsilon_i)] = X_{i,*}\ \beta,$$

where you take the expected/sample value of the terms in $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ and select ϵ to be the mean value $= 0$. The variance of this is[12]:

$$Var(y_i) = \mathbb{E}[y_i - \mathbb{E}(y_i)]^2.$$

Separating and expanding the squared terms:

$$\mathbb{E}(y_i^2) - [\mathbb{E}(y_i)]^2 = \mathbb{E}[(X_{i,*}\beta)^2 + 2\epsilon_i X_{i,*}\beta + \epsilon_i^2] - (X_{i,*}\beta)^2 = \mathbb{E}(\epsilon_i^2).$$

The mean value and variance of ϵ, where ϵ is assumed to have $N \sim (0, \sigma^2)$, can be selected for $2\epsilon_i X_{i,*}\beta$ and ϵ_i^2, respectively so that:

$$Var(\epsilon_i) = \sigma^2.$$

Hence, $y_i \sim N(\mathbf{X}_{i,*}\,\boldsymbol{\beta}, \sigma^2)$, that is $\boldsymbol{y}$ follows a normal distribution with mean value $\boldsymbol{X}\boldsymbol{\beta}$ and variance σ^2. With the OLS expressions for the optimal parameters $\hat{\boldsymbol{\beta}}$ can be shown as, where $\beta = X^T X)^{-1} X^T Y$ [12]:

$$\mathbb{E}[(\hat{\beta})] = \mathbb{E}[(X^T X)^{-1} X^T Y],$$

The expected value of X is itself and, as shown above, $\mathbb{E}[(y_i)] = X_{i,*}\ \beta$. Thus:

$$= (X^T X)^{-1} X^T \mathbb{E}(Y) = (X^T X)^{-1} X^T X\beta = \beta.$$

The variance of β is defined as [12]:

$$Var(\hat{\beta}) = \mathbb{E}[\hat{\beta} - \mathbb{E}[(\hat{\beta})]][\hat{\beta} - \mathbb{E}[(\hat{\beta})]^T.$$

$$= \mathbb{E}[(X^T X)^{-1} X^T Y - \beta][(X^T X)^{-1} X^T Y - \beta]^T.$$

Expanding and redefining the terms:

$$= (X^T X)^{-1} X^T \mathbb{E}[YY^T] X (X^T X)^{-1} - \beta\beta^T$$

$E[YY^T]$can be defined as (X $\beta\beta^T X^T + \sigma^2 I_{nn}$, where I_{nn} is the identity matrix [12]:

$$= (X^T X)^{-1} X^T (X\beta\beta^T X^T + \sigma^2 I_{nn}) X (X^T X)^{-1} - \beta\beta^T$$

Expanding and reordering terms:

$$= \beta\beta^T + \sigma^2 (X^T X)^{-1} - \beta\beta^T = \sigma^2 (X^T X)^{-1}$$