Project 1 FYS-STK4155

Mia Synnøve Frivik, Andrea Myrvang, Max Jan Willem Schuringa, Janita Ovidie Sandtrøen Willumsen
https://github.com/Mia-F/FYS_STK_Project_1.git
(Dated: October 15, 2023)

This report, crafted by Mia Synnøve Frivik, Andrea Myrvang, Max Jan Willem Schuringa, and Janita Ovidie Sandtrøen Willumsen, aims to explore the effects of different regression methods on data modeling, focusing on Ordinary Least Squares (OLS), Ridge, and LASSO. Utilizing synthetic data from the Franke function. Our analysis puts a lens on how these regression methods handle various data types. Our goal is to ascertain how each method influences predictive modeling, particularly in identifying high-risk zones for events like avalanches and floods. We'll employ bias-variance trade-off analysis and cross-validation to rigorously evaluate our models' performances and ensure their reliability. Findings indicate that the choice of regression method highly depends on the the data set used for the analysis.

I. INTRODUCTION

Machine learning is a powerful tool useful in many fields of research. One illustration of its utility is its application to terrain data analysis. Through the creation of terrain models from real data of a specific geographic area, one can effectively anticipate high-risk avalanche zones [9], potentially leading to life-saving interventions. These methods can extends to addressing concerns related to floods, which has become a hot topic this past month following the storm Hans. Machine learning can also aid in spatial planning challenges, useful in big cities all over the world. It is fair to say machine learning possesses immense potential when solving complex and relevant challenges in our modern society, encompassing climate-related issues, urban planning, and life-saving endeavors.

In this report er are going to study three different regression methods, ordinary least squares (OLS), Ridge and LASSO to see how these methods compare when applied to different data sets. First we are going to use the Franke function to make dummy data to validate if our models work. When plotted in the interval $[0, 1]$ this function looks like a mountain and a valley, which is a perfect starting point when we later want to apply these methods to real digital terrain data obtained from https://earthexplorer.usgs.gov/. To more accurately simulate the realism of practical machine learning scenarios, we will impose limitations on our data sets. Additionally, we will employ techniques such as bootstrapping and cross-validation to expand our data set size and assess their impact on model validation.

First, we will outline the methods and models employed. Then we present our results and a discussion of them, followed by the conclusion.

II. THEORY

A. Linear regression methods

Linear regression is a foundational statistical modeling technique employed to predict continuous target variables based on one or more input features. It operates under the assumption that there exists a linear relationship between the input features and the target variable. The primary objective of linear regression is to determine the coefficients to establish the optimal linear model that fits the given data the best. A linear regression model has the following form :

$$f(\mathbf{X}) = \beta_0 + \sum_{i=1}^{p} X_i \beta_i [2] \tag{1}$$

Where β is the coefficients and X the design matrix. It is important to note that this method assume that either the function is linear or approximately linear.
There exist several variants of linear regression, each of which offers unique characteristics and advantages. In this particular context, we will narrow our focus to three fundamental techniques: Ordinary Least Squares (OLS), Ridge regression and LASSO regression. These techniques will be employed in the context of analyzing the two-dimensional Franke function.

1. *Ordinary least squares (OLS)*

Ordinary Least Squares (OLS) is a fundamental technique in regression analysis, where the aim is to create a model that minimize the difference from the observed data and the predicted model. This is achieved by estimating the coefficients in a linear regression model, as defined in Equation (1) To minimize the MSE, we calculate the optimal β values. This involves taking the derivative of the cost function, as shown in Equation (2), and setting it to zero.

$$C(\beta) = \frac{1}{n} \left\{ (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \right\} \tag{2}$$

By solving $\frac{\partial C(\beta)}{\partial \beta} = 0$, we determine that for OLS, the optimal β values can be obtained by solving Equation (3).

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

While OLS is a straightforward and efficient linear regression method to implement, its limitations become particularly evident when dealing with data sets that contain substantial amounts of noise. In response to the limitations of OLS in noisy data sets, two other linear regression methods have been developed namely Ridge and LASSO regression.

2. Ridge

Ridge regression closely resembles OLS, but it has an incorporation of a penalty term. The penalty term is defined into the cost function as depicted in equation (4).

$$C(X,\beta) = \big\{(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)\big\} + \lambda\beta^T\beta \tag{4}$$

If we take the derivative of the cost function $\frac{\partial C(X,\beta)}{\partial \beta}$ and solve the equation when the derivative is zero, we then obtain the equation for the optimal β:

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{5}$$

This extra term $\lambda\mathbf{I}$ helps controlling the size of the coefficients. If λ is small, then the impact of the penalty term is minimal, resulting in coefficients similar to those obtained through OLS. On the other hand, larger values of λ lead to a more significant impact from the penalty term, resulting in smaller coefficient values.

3. LASSO

Least Absolute Shrinkage and Selection Operator (LASSO) is a shrinkage method similar to Ridge Regression. Both methods work towards minimizing the residual sum of squares while simultaneously incorporating a penalty term. The primary distinction between these regression techniques lies in the way the penalty term is defined into the cost function.
In LASSO regression, the norm-2 vector $\sum_i \boldsymbol{\beta_i^2}$, which is employed in Ridge Regression, is replaced by the norm-1 vector $\sum_i |\boldsymbol{\beta_i}|$, leading to a nonlinear solution of y_i [7]. The cost function for LASSO is shown under in equation (6).

$$C(X,\beta)_{\text{Lasso}} = \big\{(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)\big\} + \lambda||\boldsymbol{\beta}||_1 \tag{6}$$

where $||\boldsymbol{\beta}||_1 = \sum_i |\boldsymbol{\beta_i}|$ which is found by taking the derivative with respect to $\boldsymbol{\beta}$ [7],

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda sgn(\boldsymbol{\beta}) = 2\mathbf{X}^T\boldsymbol{y} \tag{7}$$

LASSO does not yield a closed-form expression due to a non linear solution, necessitating a numerical approach for its determination. This method is essential in regression analysis because it carefully balance precision and clarity by penalizing the absolute size of regression coefficients [8].

B. Model evaluation methods

Model evaluation methods assess the performance and effectiveness of a regression model. These evaluations help gauge the accuracy, goodness-of-fit and general capabilities of the models. Common methods of evaluation include calculating mean squared error (MSE), R^2-Score and Bias-variance trade off.

1. Mean Square Error

Mean Squared Error, abbreviated MSE, is a commonly used risk metric that corresponds to the expected value of the squared (quadratic) error [3]. It is defined as

$$MSE(\boldsymbol{y},\tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i\tilde{y}_i)^2 \tag{8}$$

MSE measures the average of the squared differences between predicted values and actual outcomes. Therefore, the smaller the value, the better the fit between the predictions and reality. Since the MSE value comes from the square of the errors between the actual values and the predicted values, it cannot be negative. A perfect fit between the predictions and reality would yield an MSE value equal to zero [3]. A disadvantage with MSE is that it's not scaled. Outliers will thus contribute the same amount as any other point, which can lead to an skewed MSE value.

2. R^2-Score

The R^2-score function is a popular error metric that computes the coefficient of determination, which indicates how accurately a model can predict future samples. The score typically ranges from 0 to 1, but it can also be negative. A score of 1 indicates a perfect fit, while a negative value indicates an inadequate model. A constant model that always predicts the expected value of the target variable, regardless of the input features, would receive an R^2-score of 0 [3]. The R^2-score is defined in (9)

$$R^2(\boldsymbol{y},\tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{9}$$

where we have defined the mean value of $\boldsymbol{y}$ as $\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$.

3. Bias-variance tradeoff

OLS relies on assumptions about the data. We can assume a normal distribution of the output data with mean, given by the expectation value, and variance. These properties come into play when analyzing the bias-variance trade-off of our models. For OLS we need

$$\mathbb{E}_{OLS}(y_i) = \mathbf{X}i, *\beta \tag{10}$$
$$\mathbb{E}_{OLS}(\beta) = \beta \tag{11}$$
$$\mathbb{V}_{OLS}(y_i) = \sigma^2 \tag{12}$$
$$\mathbb{V}_{OLS}(\hat{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} \tag{13}$$

Refer to A for derivation of the expressions. To assess the accuracy of our Ridge model, we need

$$\mathbb{E}(\beta_{Ridge}) = (X^TX + \lambda I)^{-1}(X^TX)\beta \tag{14}$$
$$\mathbb{V}(\beta_{Ridge}) = \sigma^2[X^TX + \lambda I]^{-1}X^TX\{[X^TX + \lambda I]^{-1}\}^T \tag{15}$$

Derivation of expression for both the optimal β and variance can be found in the appendix B. The Bias-variance trade-off is a measurement of how well a model fit the real data. We can use the expression for expected value of the mean square error as a trade-off, using equation the equation in B for a continuous function

$$\mathbb{E}((\boldsymbol{y} - \tilde{\boldsymbol{y}})^2) = \mathbb{E}((f(\boldsymbol{x}) - \mathbb{E}(\tilde{\boldsymbol{y}}))^2) + \mathbb{V}(\tilde{\boldsymbol{y}}) + \sigma^2 \tag{16}$$

The bias let us know if the model complexity allows us to describe the data, and that error is not introduced as a result of simplification. The variance let us know our model predictions vary between different data set. The ideal model has both low bias and low variance.

C. Resampling techniques

The main limitation in machine learning is the size of the data sets available to train the models. In some cases experiments are both expensive and time consuming, yet they result in a small amount of data. Additionally, there are also situations where its not possible to generate new data with experiments. In these cases it is necessary to have methods where we can generate data sets with the same properties as our original data set. Bootstrap and cross validation are two such method, where we use the data available to generate data sets.

1. Bootstrap

The bootstrap method is a re-sampling procedure, which generate several sample data sets from the original data set by random sampling with replacement [4]. That is, if we have a data set D with n data points, the elements can be represented as

$$D = d_1, d_2, d_3, d_4, \ldots, d_n \tag{17}$$

If we apply the bootstrap method on the data set, a sample output D^* can be represented as

$$D^* = d_3, d_n, d_4, d_4, \ldots, d_2 \tag{18}$$

Since we are sampling the data using replacement, one observation can appear multiple times in the new data set. If we re-sample several data points and calculate the mean value of the resulting data set

$$d_{new} = \frac{1}{k}(d_1, \ldots d_k) \tag{19}$$

we have a method of producing "new" data sets from a limited number of data points, to train our model with. We can calculate the mean and standard deviation of each data set to evaluate the model statistically [1]. One important advantage of the bootstrap method is that the data can be split in to test and train before we re-sample the data, which means the test data can be kept separate from the training model. When we test the model, the data will be unseen and give an indication of how well the model perform on real data.
When we generate a new data set we draw samples from the same training data, and calculate the mean square error of each prediction. The initial distribution of our data is not known, however, when the sample size is large the mean square error will follow a normal distribution. The Central Limit theorem [5] states that for a large number of samples the law of large numbers [6] ensures the value of the mean converges toward the sample mean. The mean square error of our models prediction will follow a normal distribution, and will be independently distributed.

2. Cross validation

Cross validation (CV), specifically k-fold cross validation, is a technique designed to assess how well a model will generalize to an independent data set. Unlike the bootstrap method, which samples with replacement, CV partitions the original data set into k distinct subsets or "folds". One of these folds is kept as the test set, while the remaining $k-1$ folds are used for training the model. This process is repeated k times, each time with a different fold reserved as the test set and the remaining folds as the training set. Given a data set D with n data points represented as

$$D = \{d_1, d_2, d_3, \ldots, d_n\} \tag{20}$$

For $k = 5$, an example partition can be given by

$$\begin{aligned}
&\text{Fold 1} = \{d_1, d_2, \ldots, d_{\frac{n}{5}}\} \\
&\text{Fold 2} = \{d_{\frac{n}{5}+1}, d_{\frac{n}{5}+2}, \ldots, d_{2\times\frac{n}{5}}\} \\
&\qquad\vdots \\
&\text{Fold 5} = \{d_{4\times\frac{n}{5}+1}, d_{4\times\frac{n}{5}+2}, \ldots, d_n\}
\end{aligned}$$

In each iteration, one of these folds is reserved for testing and the others for training, cycling through until each fold has been the test set once. The model's performance is then averaged over the k test sets to produce a single estimation. The advantage of CV is its ability to utilize the entire data set for both training and testing, which can be very helpful when data is limited. By averaging over multiple trials, CV reduces the variance associated with a single random train-test split, providing a more robust measure of model performance. However, CV can be computationally intensive, especially for large k values or large data sets. Also, unlike the bootstrap method where test data is entirely separate from training, there's an overlap in the training data across folds in CV. This means that CV may sometimes be overly optimistic about a model's performance. In the context of our study, CV aids in parameter tuning, especially for models like Ridge and LASSO where the regularization strength is crucial.

III. METHOD

A. Franke function

In the first part of this project the Franke function (21) was used as data to analyse.

$$\begin{aligned} f(x,y) = &\frac{3}{4}\,exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\,exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\,exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{21}$$

A graphic representation is shown in figure III A The

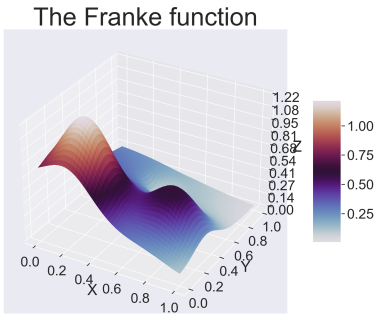

Figure 1. A plot of the Franke function

number of data points created of the Franke function was a 20×20 matrix where 20% of the data set was set aside as testing data while the remaining 80% was used for training. The noise was reduced from $\mathcal{N}(0,1)$ as proposed in the project description, to $\mathcal{N}(0,0.1)$ for better results.

1. *OLS*

First the data was fitted with the OLS method, where varying degrees of polynomials was used to create the design matrix. Since the design matrix in this case was non-invertible, singular value decomposition was used to find the β-values needed to create a model. After the regression was done, the mean square error and the R2 score was calculated for both the testing and training data sets. The different coefficients and β values was plotted for polynomial degrees in range $0-5$. The result for this analysis of the Franke function is shown in figures (3), (4) and (5).
Next the bootstrap method was implemented with 100 iterations to see how this affected the result of the OLS regression analysis. This method was implemented after the data set was split into a train and test set, to keep test data separate when finding β values. After re-sampling the training data, the model was trained. Mean square error was calculated for each training and prediction, in addition to the bias and variance, for polynomial degree $0-15$. The error of the training data was plotted against the error of test data, and the error of test was plotted against the bias and variance of the model.

2. *Ridge*

Next Ridge regression was used on the Franke function, to see if this methods has a better fit than what was obtained with OLS. Different values for λ was used to obtain the best fit as possible for each polynomial degree. For Ridge equation (5) was used to calculate the coefficients. The λ was put to zero to see if the model became the same as for OLS, lastly the β values for different orders of polynomial was used to fit the models. The models was then plotted to see how these varied and compered to those from the OLS regression.
In the pursuit of assessing the robustness and reliability of Ridge regression models, we employed cross-validation, specifically scrutinizing the impact of varying regularization parameters, denoted as λ. The function `k_fold`, was developed to execute the k-fold cross-validation technique, accepting the data set and an integer k as arguments, and subsequently partitioning the data into k randomized subsets (folds). It returns k pairs of training and test indices, each representing a distinct division of the data, enabling model evaluation across varied data scenarios.
For each value of λ, increased with a factor of 10, the Ridge regression model was trained and validated k times - once per fold. The data was divided into train and test sets. The design matrix was generated from the

x and y values, using a specified max polynomial degree. The Ridge model, instantiated with the current λ, was then fitted with training data, and predictions were made. The Mean Squared Error (MSE) between the predictions and actual test values was computed and stored in a scores array. The MSE values were averaged per λ, providing an unbiased performance metric, and facilitating the analysis and visualization of how distinct regularization parameters influenced model performance.

3. *LASSO*

The LASSO regression equation doesn't result in a straightforward analytical solution like Ridge regression or ordinary least squares. Sci-kit learn was used to help in these computations, and the LASSO regression function was implemented from it's linear regression package. Next, cross validation was applied to LASSO regression using the same method as employed for Ridge regression.

B. Real terrain data

In the last part of this project real terrain data was analysed. Due to the massive size of the terrain data only the first $500 \cdot 500$ matrix as shown in figure 2 was used to avoid problems with computer memory. For LASSO we had to use a $20 \cdot 20$ data matrix to be able to run the code in a acceptable amount of time.

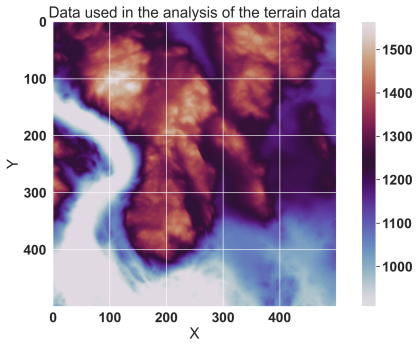

Figure 2. A plot of the data matrix used in the analysis of the terrain data in the second part of project 1.

The data was also scaled by first subtracting the mean value from each column and then divide on the standard deviation so that the variation in height in our data set would not lead to huge MSE values. The OLS regression was applied to create a model of the data set fitted with polynomials of degree $0 - 10$. The same was done for Ridge, but here also a heat map of the MSE as a function of both different λ values and complexity was created. The same was done for LASSO but here as mentioned above one had to downsize the data set to a $20 \cdot 20$ matrix when running over different λ values, for the calculation of MSE and R2 score as a function of complexity the normal size data was used ($500 \cdot 500$ matrix).

IV. RESULTS

A. Franke function

1. *OLS*

Our initial step involved the application of OLS to the Franke function without noise. This analysis was conducted without employing any resampling techniques, and our data set consisted of $20 \cdot 20$ data points. The results for the MSE and R2 score is shown in figure (3)

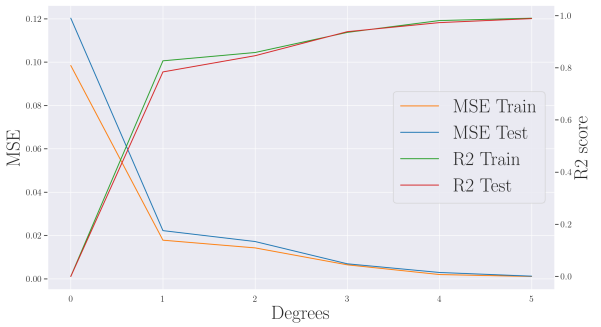

Figure 3. Plot showing the MSE and R2 score for the Franke function without noise. The regression method used was OLS.

The next step was to include noise given by the normal distribution $\mathcal{N}(0, 0.1)$. Figure (4) shows how the MSE and R2 score changes when noise is included in to the data set.

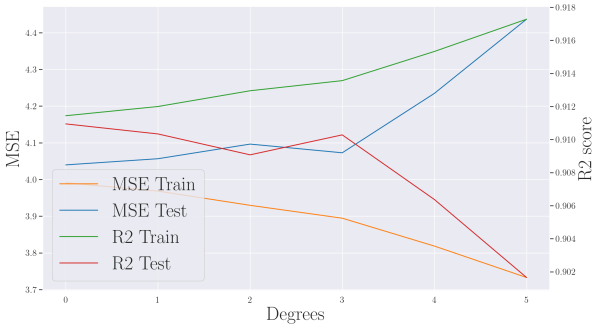

Figure 4. Plot showing the MSE and R2 score for the Franke function with noise $\mathcal{N}(0, 0.1)$. The regression method used was OLS.

We then plotted the coefficients for the different orders of polynomials to see how these varies in value.

Figure 5. A plot showing the β values for OLS for different orders of polynomials.

Figure 6 is showing the error of training and test data for polynomial degree $0-15$. There is a consistent reduction in error of training for higher polynomial degree, whereas the error of test increase with polynomial degree $6-15$.

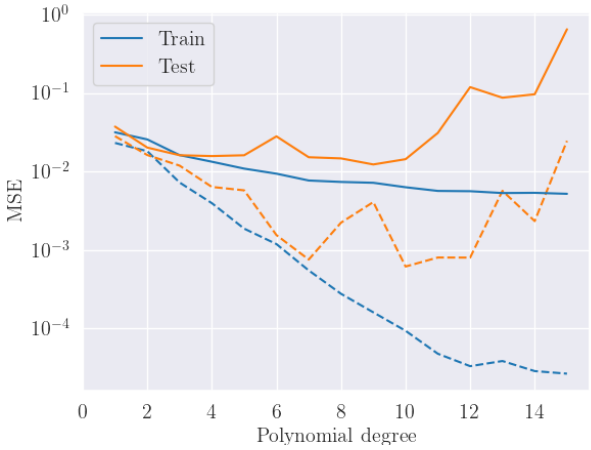

Figure 6. MSE of the models prediction for polynomial degree $0-15$, using the Franke function. The solid lines describe the MSE where noise is added to the function, dotted lines describe MSE where noise is not added.

Figure 7 is showing the error of prediction, and the bias and variance of the model for polynomial degree $0-15$.

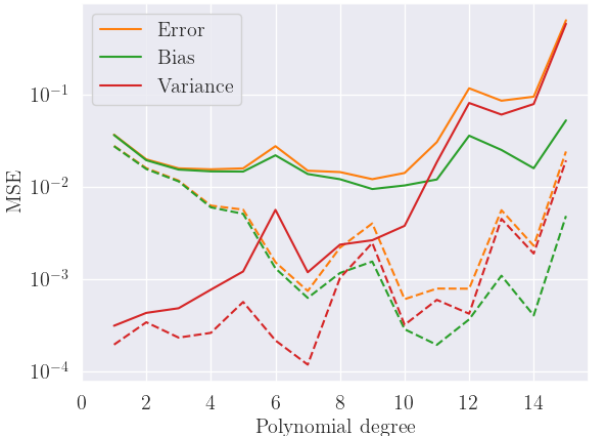

Figure 7. MSE of the models predicted error, using the Franke function, bias and variance. The solid lines describe the MSE where noise is added to the function, dotted lines describe MSE where noise is not added.

We compared number of folds used when re-sampling data from the Franke funktion without noise, by using OLS as regression method. In figure 8 we plot MSE for each fold $k = [5-10]$, to determine an ideal division of the data available.

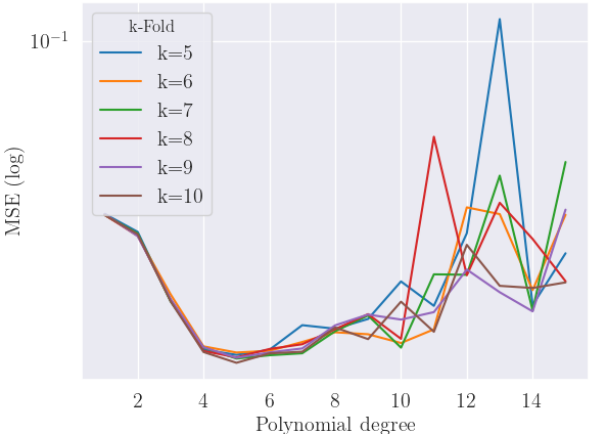

Figure 8. MSE of prediction error for polynomial degree $0-15$, using the Franke function, with cross validation $k = 5-10$.

2. *Ridge*

For Ridge regression we started by making a heat map of how the MSE changes as a function of complexity and λ values. Here we tested for 1000 different *lambda* values from $\lambda_0 = 10^{-8}$ to $\lambda_{1000} = 10^2$ and for polynomials with degree 1-5. A plot of the heat map for both the training and testing data set is shown in figure (9) and (10).

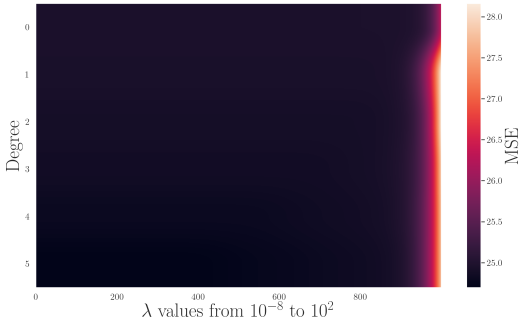

Figure 9. A heat map of the MSE for the training data, for different λ values and complexities. The λ values goes from 10^{-8} to 10^{2}.

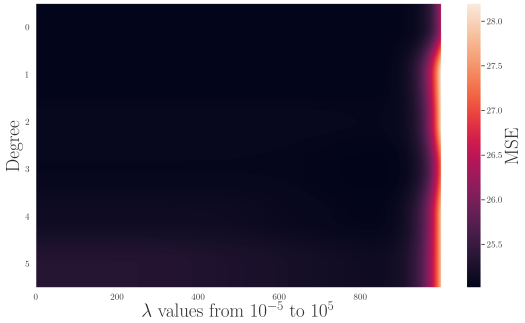

Figure 10. A heat map of the MSE for the testing data, for different λ values and complexities. The λ values goes from 10^{-5} to 10^{5}.

Next we used the first λ to plot how the MSE and R2 score changes as a function of complexity for the Franke function both with and without noise. Figure (11) shows the MSE and R2 score with no noise present in the data set of the Franke function, and figure (12) shows with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

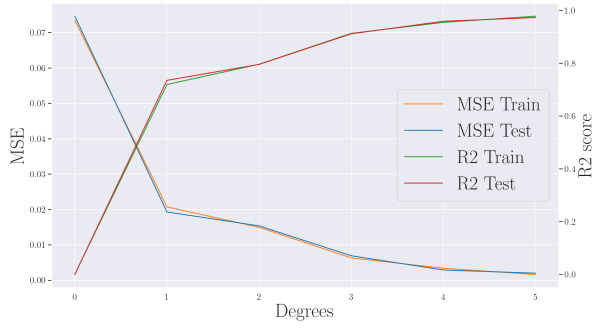

Figure 11. A plot of the MSE and R2 score as a function of degrees and a λ value of 10^{-5} for Ridge regression on the Franke function with no noise present.

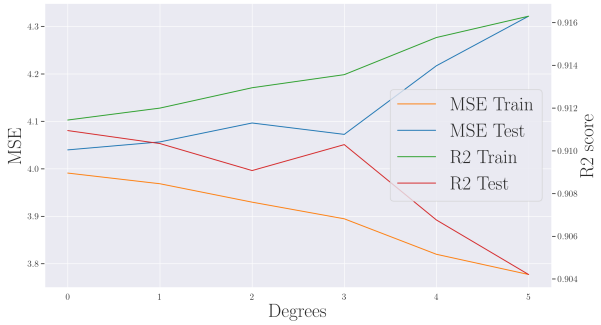

Figure 12. A plot of the MSE and R2 score as a function of degrees and a λ value of 10^{-5} for Ridge regression on the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

We can also see what happens for the ridge regression if λ is put to zero. Figure (13) shows the MSE and R2 score if $\lambda = 0$, while figure (14) shows the MSE if $\lambda = 0$ and $\lambda = 10^{-5}$.

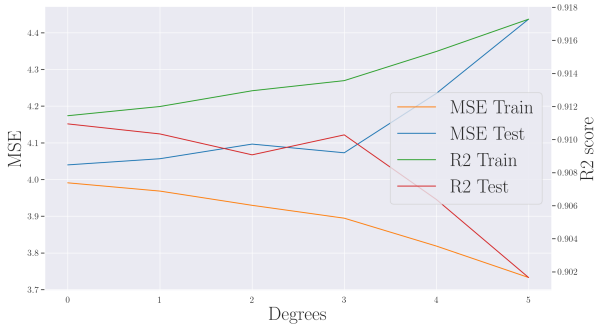

Figure 13. A plot of the MSE and R2 score as a function of degrees and a λ value of 0 for Ridge regression on the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

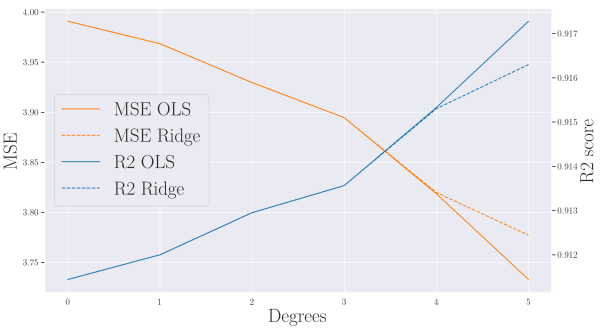

Figure 14. A plot of Ridge regression were solid lines shows when $\lambda = 0$, so the same as OLS and the one with doted lines is for $\lambda = 10^{-5}$.

In Figure (15) the Mean Squared Error (MSE) is depicted on the y-axis, with the logarithm to the base 10 of the hyper parameter λ ($\log_{10}(\lambda)$). For the x-axis we set the polynomial degrees ranging from 0 - 15. Different lines in varying colors are plotted to showcase the MSE across different λ values.

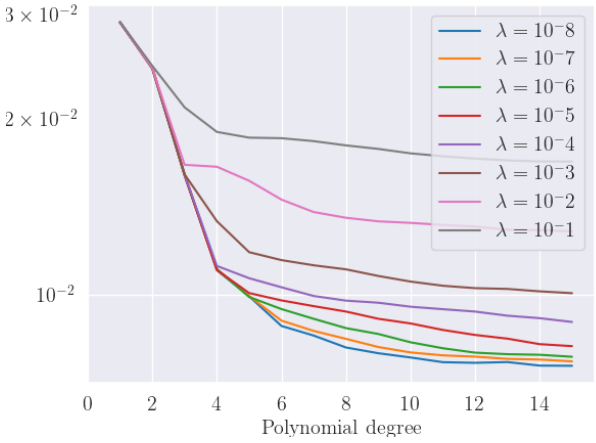

Figure 15. MSE of ridge prediction error for polynomial degree $0 - 15$, using the Franke function, with cross validation $k = 5$.

Lastly for Ridge regression we plot the β values for different complexities in figure (16).

Figure 16. A plot showing the β values for OLS for different orders of polynomials.

3. *LASSO*

For LASSO regression we started by making heat maps for both the training data (figure (17)) and test data (figure (18)) with the same λ values (10^{-8} to 10^{2}) and polynomial degrees (1-5) like for Ridge regression .

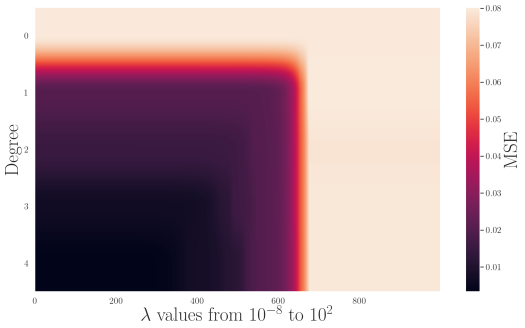

Figure 17. A heat map of the MSE for the training data, for different λ values and complexities. The λ values goes from 10^{-8} to 10^{2}.

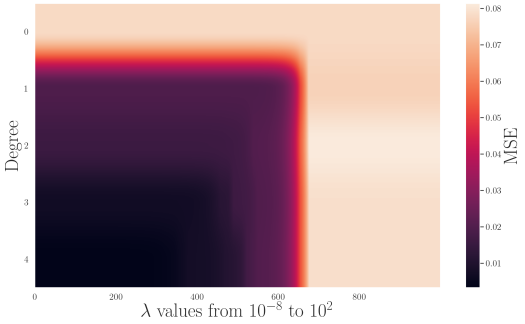

Figure 18. A heat map of the MSE for the testing data, for different λ values and complexities. The λ values goes from 10^{-8} to 10^{2}.

We then plotted the MSE and R2 score as a function of complexity both without noise (figure (19)) and with noise given by the normal distribution $\mathcal{N}(0, 0.1)$ (figure (20)) for $\lambda = 10^{-5}$.

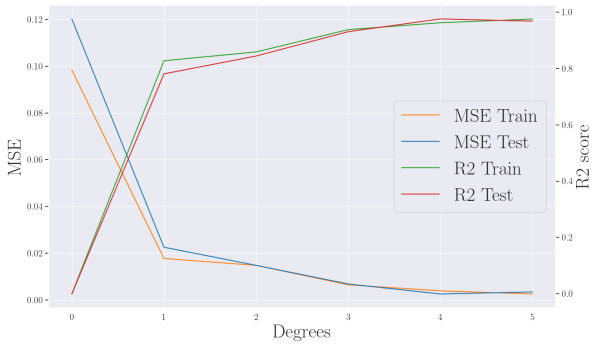

Figure 19. A plot of the MSE and R2 score as a function of degrees and a λ value of 10^{-5} for LASSO regression on the Franke function with no noise present.

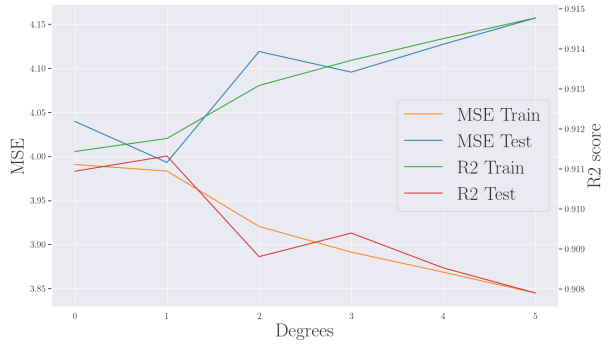

Figure 20. A plot of the MSE and R2 score as a function of degrees and a λ value of 10^{-5} for LASSO regression on the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

We then plotted the MSE for OLS, Ridge ($\lambda = 10^{-5}$) and LASSO ($\lambda = 10^{-5}$) given in figure (21) .

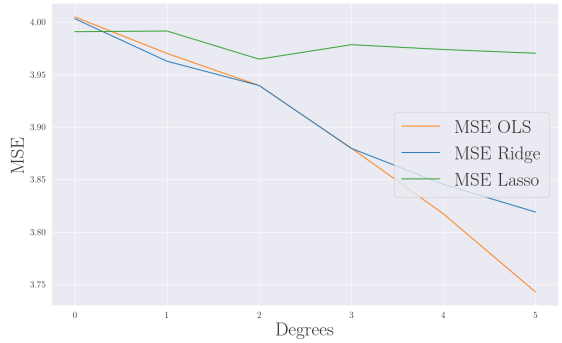

Figure 21. A plot of OLS, Ridge regression with $\lambda = 10^{-5}$ and LASSO regression with $\lambda = 10^{-5}$ on the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

In Figure (22), the Mean Squared Error (MSE) is plotted on the y-axis in a logarithmic scale. The x-axis represents the polynomial degrees, which range from 0 to 14. Multiple colored lines are shown, each representing a specific value of the regularization parameter λ. These lines provide a visual representation of the MSE across different λ values for varying polynomial degrees, assisting in the determination of an optimal polynomial degree and regularization parameter for LASSO regression.

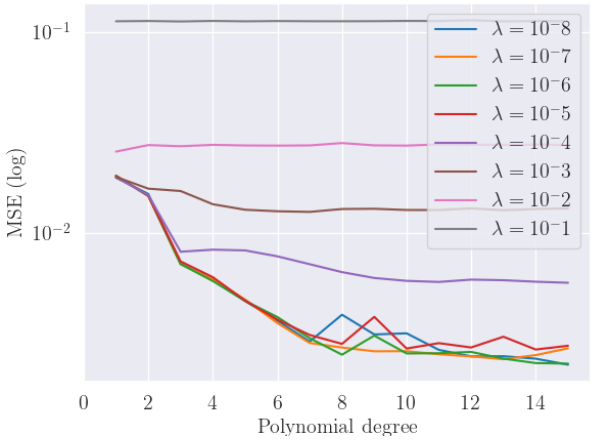

Figure 22. MSE of LASSO prediction error for polynomial degree $0-15$, using the Franke function, with cross validation $k = 5$.

B. Real terrain data

1. *OLS*

Now we look at the analysis of the real terrain data. First we performed a OLS regression on the data set, where we varied the complexity between 0 and 10 degrees. The MSE and R2 score is shown in figure (23).

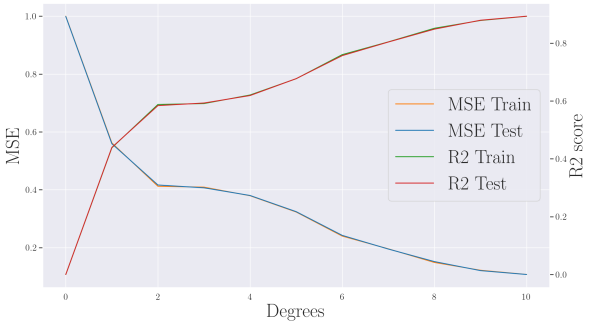

Figure 23. A plot showing the MSE and R2 score for the OLS regression on the terrain data.

We compared re-sampling techniques for the terrain data, using OLS as regression method. MSE for both are shown in figure 24.

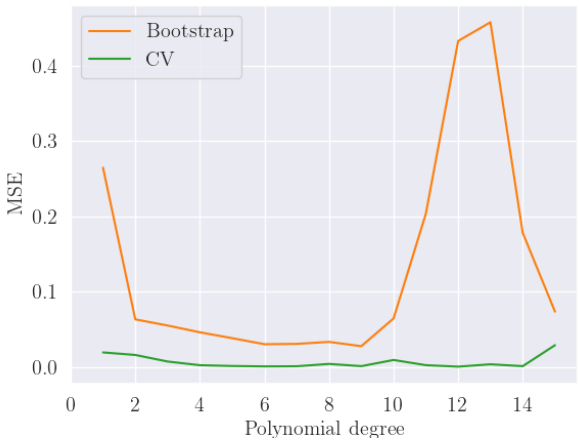

Figure 24. MSE as a function of complexity for OLS with terrain data, using both bootstrap with 100 samples, and cross validation with $k = 5$.

2. *Ridge*

Next we implemented Ridge regression on the terrain data. We made a heat map of the MSE values as a function of λ values and complexity. The complexity varied from 1 to 10 and the λ values varied from 10^{-8} to 10^{2}. The result is shown in figure (25) and (26).

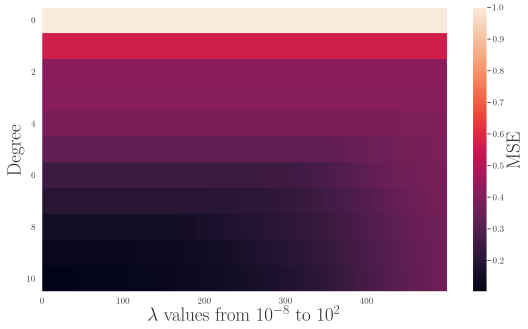

Figure 25. Heat map of the MSE for the training data, as a function of complexity and *lambda* values.

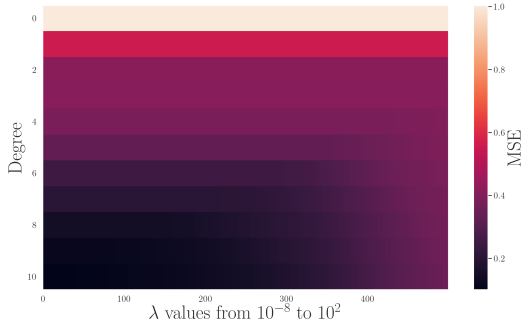

Figure 26. Heat map of the MSE for the test data, as a function of complexity and *lambda* values.

We also plotted the MSE and R2 score as a function of only complexity, where λ was put to $\lambda = 10^{-5}$. The result are shown in figure (27).

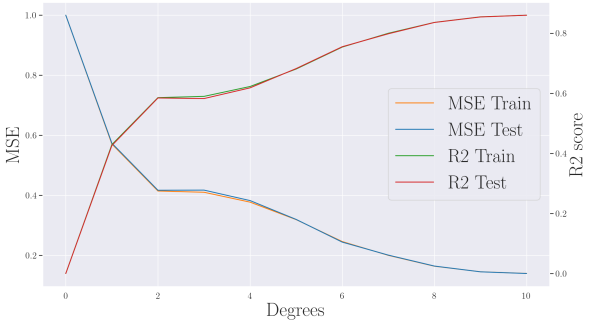

Figure 27. MSE and R2 score as a function of complexity with $\lambda = 10^{-5}$ for the terrain data modelled with Ridge regression.

We then plotted the MSE for both OLS and Ridge with a lambda value of 10^{-5} in the same plot to see how these compered to each other. The plot is shown in figure (28)

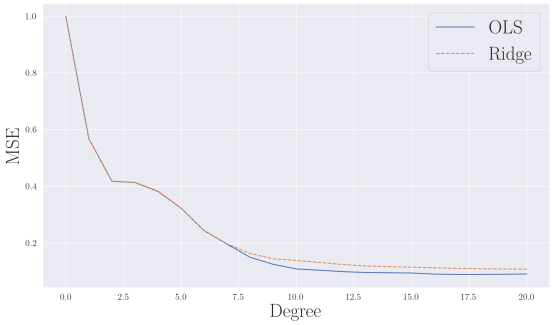

Figure 28. MSE as a function of complexity for both OLS and Ridge were Ridge is implemented with a λ value of 10^{-5}.

The plot provided in figure (29) showcases the performance of Ridge Regression applied to real terrain

data across various polynomial degrees and regularization strengths (denoted by λ).

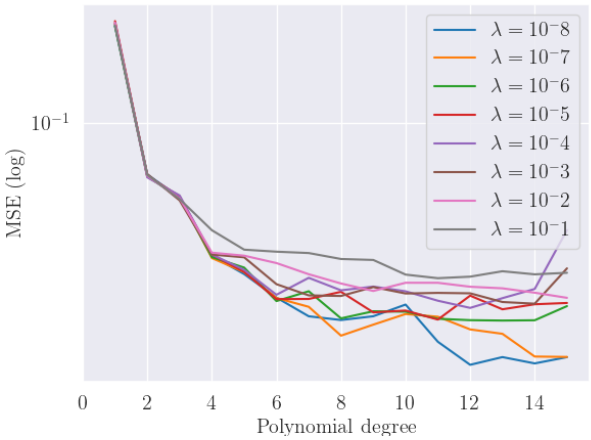

Figure 29. MSE as a function of complexity for Ridge with terrain data, using cross validation with $k = 5$.

3. *LASSO*

Last we applied LASSO regression on the terrain data. Due to the time used to run this code the heat map is made out of 10 λ values instead of 1000 as for Ridge. The heat map for both the training data and for test data is shown in figure (30) and (31).

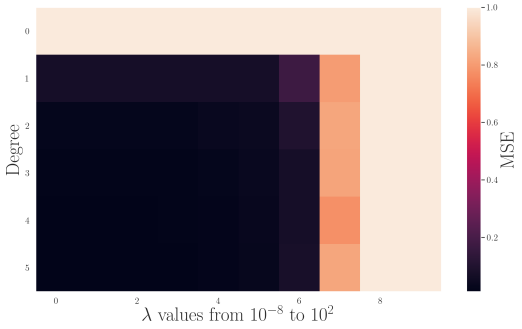

Figure 30. Heat map of the MSE for the training data, as a function of complexity and *lambda* values for LASSO regression.

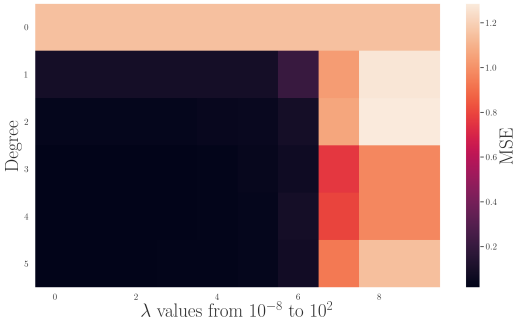

Figure 31. Heat map of the MSE for the test data, as a function of complexity and *lambda* values for LASSO regression.

We then also plotted the MSE and R2 score as a function of only complexity, where λ was put to $\lambda = 10^{-5}$. The result are shown in figure (32).

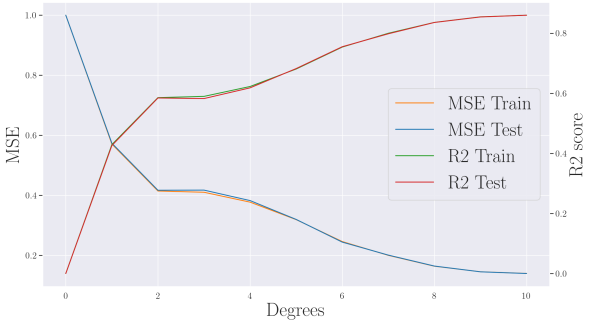

Figure 32. MSE and R2 score as a function of complexity with $\lambda = 10^{-5}$ for the terrain data modelled with LASSO regression.

We then plotted the MSE as a function of complexity for both OLS, Ridge and LASSO to compere the three regression methods for the terrain data. The plot is shown in figure (33).

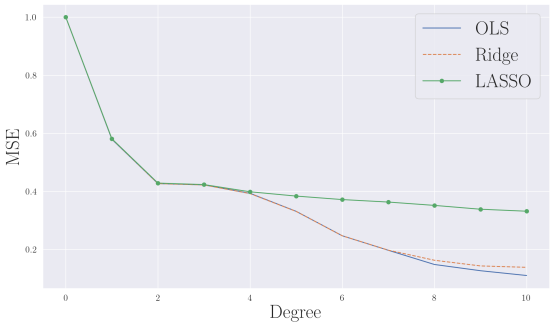

Figure 33. MSE as a function of complexity for both OLS, Ridge and LASSO were Ridge and LASSO are implemented with a λ value of 10^{-5}.

Lastly the plotted in figure (34) showcases the performance of LASSO Regression applied to real terrain data across various polynomial degrees and regularization strengths (denoted by λ).

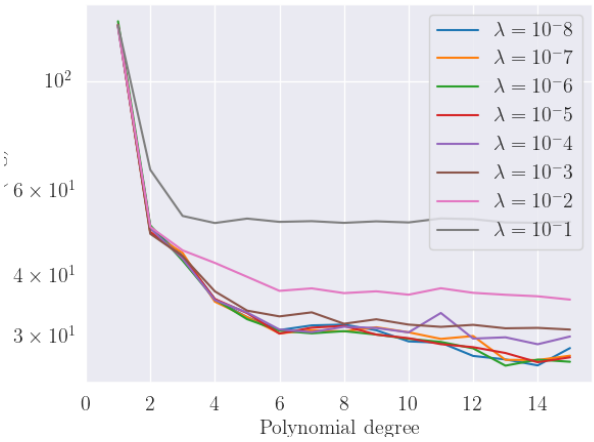

Figure 34. MSE as a function of complexity for LASSO with terrain data, using cross validation with $k = 5$.

V. DISCUSSION

A. The Franke function

It was stated in the project description that we should use a noise that was given by the normal distribution $\mathcal{N}(0, 1)$. What we found was that since the Franke Function when plotted for x and y in the range between 0 and 1 the maximal value of the Franke function becomes around 1.4. When plotted with noise that had values between 0 and 1 the plot of the Franke function becomes unrecognizable as shown in figure (35).

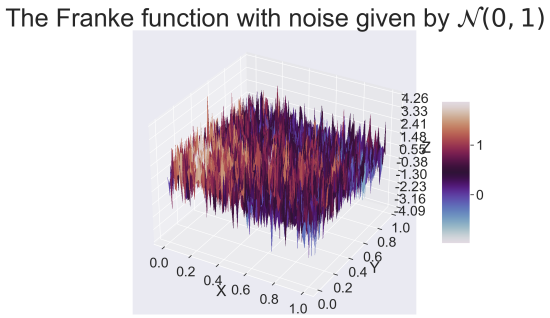

Figure 35. Plot showing the Franke function with noise given by the normal distribution $\mathcal{N}(0, 1)$.

But if instead the normal distribution $\mathcal{N}(0, 0.1)$ is used, the data set becomes much easier to work with, and the results becomes much prettier (as shown in figure (36)) since noise has a tendency of ruining everything.

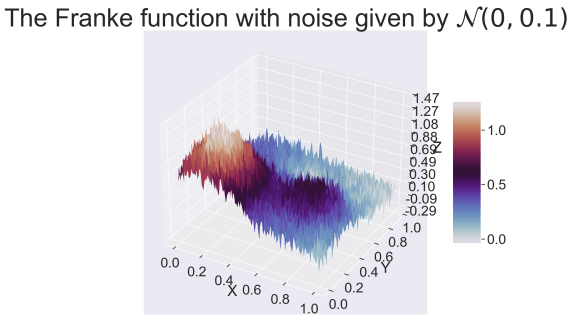

Figure 36. Plot showing the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$.

Now that we have gotten that out of the way, we can start actually discussing the result of the regression analysis of the Franke function.

B. The Franke function

1. *OLS*

We start with the analyses of the OLS implementation on the Franke function. It is important to note that since the parameter x and y only had values from 0 to 1 when creating the Franke function, we did not see it necessary to scale the data for any of the analysis of the Franke function, since it in a way already is scaled. The first results are the OLS model of the Franke function without noise. Figure (3) shows the MSE and R2 scores as a function of the polynomial degree used to create the design matrix. The size of the data set analysis was a $20 \cdot 20$ matrix, were 20% was put aside as test data, and the rest was used for training. What we can see from the figure is that the MSE is larger for lower polynomial degrees and starts to lower with higher values. At a fifth order polynomial degree the MSE approaches zero both for the test and training set. This is as expected, since there is no noise present there will not be any possibility for overfitting the model to the noise. Therefore the MSE will become lower and lower for higher order polynomials as shown in figure (3). When it comes to the R2 score we see that it becomes closer and closer to one with higher order polynomials. This is due to our model becoming increasingly better with higher orders as shown form the MSE values.
Next we look at what happens when we introduce noise given by the normal distribution $\mathcal{N}(0, 0.1)$. Figure (4) show the MSE and the R2 score for this case. Here we can see a classical example of overfitting the model to the noise. We see that the MSE becomes better and better for the training set but for the test set we see a sudden

increase after fitting with a fourth order polynomial. This is due to the amount of data points in the different data sets, since the test data set only have 20% of the original data while the training set consist of the remaining 80%. If we increase the number of data points the predicted model for the test data will converge towards that of the training model. From figure (3) we see that when our data set does not include noise the MSE for the training data gets lower with increase in complexity, while the R2 score gets closer and closer to 1. But when we introduce noise we see from figure (4) that for our test data the MSE actually increase with higher complexity.

Next we can look at the β values for the OLS models. What we can see from figure (5) is that for the higher order polynomials the values for the coefficients varies a lot in value. This is due to the regression method trying to fit the function to the noise, so this is a direct effect of overfitting the function with a to high order polynomial. From figure (5) we see that the beta values start to vary at as low as a fourth order polynomial and that for a fifth order polynomial the variance start to become substantially large. This is also supported by the MSE plot shown in figure (4), where we can see that the MSE starts to gradually increase after the third order polynomial. So we clearly start to move in to the over-fit area.

From the bootstrap results in figure 6 we can see a decrease in MSE up to polynomial degree 7, for both the training and test data. However, we see a increase in MSE for prediction based on test data, which indicate that the model is over-fit to the training data. When we plot bias and variance, with the test error, in figure 7 we can see how the error decrease when the bias decrease. In addition we see increase in error when the variance increase. A decrease in the bias of the model suggest a better fit, where we reach a level of complexity to fit the data given. We also see an increase in variance for polynomial degree $7 - 15$, suggesting an increase in variance when we try to explain our data with higher polynomial order than necessary.

In figure 7 we compare the error for the Franke function, both with and without noise, with the estimated bias and variance. What we see is that MSE of the model bias follow the error up to a polynomial degree of 3 for the model with noise, and is close in value up to degree 10. At degree 10 we can see the error is close to the variance. This is expected as the error of the model can be found by summing the bias and variance (**??**). At the degree where both the error and variance increase, the model has become overfitted. Meaning we are training our model using unnecessary orders of polynomial degree, resulting in predictions varying. When we compare the error with the model bias and variance for the function without noise, the relation become clearer. Here the error and variance decrease more for degrees up to 7, where the bias keep decreasing and the error and variance fluctuates, and increase from degree 12. The fluctuation in MSE for the function without noise can be due to the number of samples used being small, resulting in a small data set. However, an increase in number of data points would require higher order of model complexity to fit the data. This could be studied closer by changing seed used in generating random data points.

In figure 8 the performance of an Ordinary Least Squares (OLS) regression model across varying polynomial degrees and under different k values for k-fold cross-validation. For lower polynomial degrees (around 2 to 5), the MSE remains relatively high for all k values. This suggests that a simple linear or low-degree polynomial model may not be sufficient to capture the complexity of the data. Different k values exhibit varying performance, but they all follow a similar trend across polynomial degrees. This consistency across different k values reinforces the insights drawn about the optimal polynomial degree.

2. Ridge

For the Ridge regression we also have to take in to account which λ values to use in our model. To see how this value affect out model we have plotted a heat map were the MSE is plotted as a function of both different λ values and complexity. In figure (9) we see the heat map of the MSE for the training data set. From this figure we can see that for low λ values ($\lambda < 1$) the MSE is a bit below 4 which is a better result than what we got from OLS. We also see that if we increase λ the MSE also starts to increase to about 10 for $\lambda = 10^2$. If we now look at a plot of how the MSE changes as a function of complexity when $\lambda = 10^{-5}$ and $\lambda = 0$, so what we do is basically compere Ridge regression done with a low λ value and OLS. We see from figure (14) that The MSE error for models fitted with polynomial of degree $0, 1, 2, 3$ and 4 its the same but for a model fitted with a polynomial degree of 5 the MSE is lower for Ridge regression than for OLS. This is as we expect since the extra term in the calculation for the coefficient in Ridge will try to stop the beta values from varying a lot. We can see this is the case if we look at figure (16) were we still see that the β values for a fit with a fifth order polynomial still varies a lot compered to lower complexity's, but it is a fair bit lower than that of OLS regression. From this we can see that if λ is chosen correctly the model from Ridge regression is slightly better then that of OLS.

In our analysis with Ridge regression in Figure (15), our primary objective was to understand the model's behavior concerning varying polynomial degrees and different regularization parameters. We employed bootstrap resampling and cross-validation to ensure robust model evaluation and to minimize the bias-variance trade-off.

The plot showcases the model's Mean Squared Error (MSE) as a function of polynomial degrees ranging from 1 to 15. We experimented with various λ values to discern the impact of different regularization levels. Polynomial degree denotes model complexity. As it increases, the

model becomes capable of capturing intricate data patterns, but at the risk of overfitting. Our results highlight this delicate balance. At lower degrees, irrespective of regularization, the model tends to under perform, likely underfitting the data. However, for degrees between 6 to 10, a noticeable dip in the MSE is observed for certain λ values, suggesting an optimal complexity for this data set. These insights underscore the importance of judiciously selecting the model's complexity and regularization strength to ensure balanced and optimal performance.

3. LASSO

For LASSO regression, we did a similar analysis to that of ridge regression. The heat maps (Figure (18) and (17)) shows the Mean Squared Error (MSE) as a function of different λ values and complexity's, illustrating how the λ value affects our model. These heat maps demonstrate that a low λ value ($\lambda < 0.1$) and a high complexity yield a better result than a high λ value ($\lambda > 0.1$) and a low complexity. The optimal MSE result in the heat maps occurs for $\lambda = 10^{-8}$ and degree 4, after which it gradually worsens for higher λ values and lower complexity's until reaching $\lambda = 10^{1}$ and degree 0. In comparison to the heat maps generated for Ridge regression, the complexity appears to have significantly bigger impact on the MSE results for LASSO regression than for Ridge regression. If we examine the line map for MSE and R^2 without noise (figure (19)), we observe that LASSO with a λ value of 10^{-5} produces nearly identical results to the Ridge regression and OLS. However, if we look at figure (21) where we plot the OLS, Ridge and LASSO when noise is added we see that the MSE for LASSO is generally allot worse compared to the two others. The differences between Ridge and LASSO regressions can most likely be attributed to the limited number of iterations employed. LASSO regression runs with only a small number of iterations because of the long run time. The small amount of iterations may therefore lead to a bigger tolerance that we converge against. With more iterations added, we may have achieved a better MSE for LASSO regression. From this we can see that with noise added to he Franke function, LASSO regression performs poorly compared to Ridge regression and OLS.

In our exploration using LASSO regression, we are looking at the relationship between the model's behavior, polynomial degrees, and varying levels of the regularization parameter as denoted in Figure (22). Similar to our approach with Ridge regression, we employed techniques like bootstrap re-sampling and cross-validation to provide a comprehensive evaluation and to mitigate the effects of bias and variance in our model.

The depicted plot showcases the Mean Squared Error (MSE) as a function of polynomial degrees, spanning from 1 to 15, under diverse regularization strengths (λ values). The choice of λ in LASSO regression is critical. It not only regularizes the model but can drive certain feature coefficients to absolute zero, effectively performing feature selection.

From our results, it's evident that for extremely low regularization strengths, specifically $\lambda = 10^-8$ and $\lambda = 10^-6$, the MSE remains consistently high across all polynomial degrees, implying that the model may not be sufficiently regularized and might be overfitting to the training data. As we progress to moderate regularization strengths, particularly $\lambda = 10^-4$ and $\lambda = 10^-2$, the MSE demonstrates a dip, notably around polynomial degrees 6 to 10, suggesting these might be optimal model complexities for this data set under the given regularization. The MSE for higher regularization strengths like $\lambda = 10^0$ and $\lambda = 10^2$ seems to flatten out, indicating that excessive regularization might be nullifying the effect of added model complexity, leading to underfitting.

C. Real terrain data

1. OLS

We started by implementing OLS regression on the real terrain data. Figure (23) shows the MSE and R2 score as a function of complexity. As we can see from this figure the MSE is low, this is due to the way we have scaled the data set, Without proper scaling, the MSE would have been considerably higher due to the way our data set looks. Due to this scaling the result for each of out regression methods will not differ that much. What we see from figure (23) is as expected since the MSE decreases with complexity. As Figure (23) illustrates, our observations align with expectations: the MSE tends to decrease as model complexity increases. Notably, for a polynomial fit with a degree of 10, the MSE approaches zero, suggesting that a polynomial of this degree provides a accurate approximation of the true terrain data. This is again validated by looking at the R2 score, we see that when the complexity increases the R2 score begins to converge toward a value of one, which is expected when the MSE start to get closer and closer to zero. In the case with the terrain data we will continuously see that the results for our training and testing sets are almost identical, this is due to our data set being much larger than that of the Franke function, so the size although still being 20% of the original data set, it includes a substantial number of data points, causing the model's performance to closely resemble that of the training set.

For the comparison with cross validation and bootstrap we can see in Figure (24) that using cross validation as a re-sampling method we get a lower MSE. At lower polynomial degrees (around 2 to 4), both methods perform similarly with minimal MSE. However, as the polynomial degree increases to around 6, the MSE for bootstrap begins to rise sharply, indicating potential overfitting. In contrast, CV's MSE remains relatively stable until around degree 10, where it also exhibits a sig-

nificant increase. This suggests that CV might be more robust against overfitting for intermediate polynomial degrees. At very high degrees (12-14), both methods show a pronounced spike in MSE, highlighting the dangers of over-complicating the model.

2. Ridge

The next part of the project was to implement Ridge regression on the terrain data. Here we started by plotting a heat map for both the training and testing data set as shown in figure (25) and (26) Here we see as expected that lower λ values and higher complexity's give a better MSE values than for lower complexity's and higher λ values. We can also look at how the MSE and R2 score changes as a function of complexity for a λ value equal to 10^{-5} as shown in figure (27). Here we see a almost identical result as for OLS this has to do with the low λ value that approaches zero. This can also indicate that there is no noise too overfit the model with in the data set since we saw from the Franke function that when no noise was present the OLS regression gave the best result but for polynomials with higher orders the model tried fitting to the noise, which may indicate that there is no substantial amount of noise present in the data set after scaling. We can further validate this by looking at the LASSO implementation.
The Figure (29) illustrates the performance of Ridge Regression on real terrain data, evaluated using cross-validation, as a function of various polynomial degrees and regularization parameters (λ). At the lower end of the polynomial degree spectrum (around 0 to 2), the Mean Squared Error (MSE) is quite pronounced, hinting that simpler models might not adeptly represent the nuances of the terrain data. There is a reduction in MSE as we move towards polynomial degrees of 4 to 6. This zone seems to offer a more accurate representation of the data without excessive complexity. The region around polynomial degrees of 4 to 6 combined with λ values ranging from 10^{-5} to 10^{-3} appears to deliver the lowest MSE, marking it as a potential optimal setting for this data set.

3. LASSO

We then implemented the LASSO regression on the terrain data. Here we also plotted heat maps for both the training and test data shown in figure (30) and (31). The heat maps gave approximately the same results for the terrain data as for the Franke function. They demonstrate that lower λ values and higher complexity's gives the best MSE values as expected.
If we then examine the line map for MSE and R^2 score changes as a function of complexity for a λ value equal to 10^{-5} shown in figure (32), we see it is nearly identical to the results from the Ridge regression and OLS results. Based on the results for LASSO on the Franke function it may indicate that the similarity's between the results stem from there being little to no noise present in the terrain data set after scaling.

If we look at figure (33) where we plot the OLS, Ridge and LASSO for the terrain data, we see that all the three methods gets better with higher complicity's, but LASSO is generally allot worse compared to the two others. The LASSO regression for the real data may also be worse because of she small amount of iterations that may lead to a bigger tolerance that we converge against.

The Figure (34) define the performance of LASSO Regression on real terrain data, evaluated with cross-validation, based on various polynomial degrees and regularization parameters (λ). At the initial polynomial degrees (around 0 to 2), the Mean Squared Error (MSE) is relatively high. This suggests that models of low complexity may not adequately encapsulate the intricacies present in the terrain data. The optimal region, in terms of minimizing MSE, appears to be around polynomial degrees of 4 to 6, paired with λ values between 10^{-5} and 10^{-3}. This suggests that within these parameter settings, the model performs best for the given data set.

VI. CONCLUSION

To summarize our findings, it is clear that the choice of the best regression method depends on the data set used. In scenarios where the data is noise-free, OLS proves to be the most effective regression method. But noise-free data is usually not the case. What we found was that the best regression method for the Franke function with noise given by the normal distribution $\mathcal{N}(0, 0.1)$, was Ridge for higher polynomial degrees, this is due to OLS over-fitting the data to the noise for higher polynomial degrees, while Ridge and LASSO tries to minimise the variance in β values to avoid this problem. Even though LASSO aims to minimize variance in β values, it exhibited the worst performance among the three regression methods. However, it remains uncertain whether Lasso could outperform Ridge and OLS in handling noise due to limitations in iterations. The decision to restrict iterations was influenced by run time constraints, leaving the true capabilities of Lasso in this scenario unknown. For our terrain data we surprisingly found that OLS gave the model with the lowest MSE. From what we can see in figure (39) the terrain data looks a little noise, but it may be due to the scaling of the data set that makes OLS the best regression method for the terrain data.

The improvement potential for this project is high, due to some grope problems we got started on the project late together as one group. Therefor the structure on GitHub and the cods are quite messy. This is something we will work on improving for the next project.

VII. ACKNOWLEDGEMENT

We would like to thank Morten Hjorth-Jensen for his amazing and well crafted jupyter notebooks. Additionally, we would like to acknowledge the valuable inspiration provided by ChatGPT throughout the report-writing process.

Appendix A: Mean values and variance

The main regression method used in this report is the ordinary least squares method. This appensix shows the calculations for some of the equations used to produce the results shown in this report. We have assumed that our data can be described by the continous function $f(\boldsymbol{x})$, and an error term $\boldsymbol{\epsilon}$ $N(0, \sigma^2)$. If we approximate the function with the solution derived from a model $\tilde{\boldsymbol{y}} = X\boldsymbol{\beta}$ the data can be described with $\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$. The expectation value

$$\begin{aligned}
\mathbb{E}(\boldsymbol{y}) &= \mathbb{E}(X\boldsymbol{\beta} + \boldsymbol{\epsilon}) \\
&= \mathbb{E}(X\boldsymbol{\beta}) + \mathbb{E}(\boldsymbol{\epsilon}) && \text{where the expected value } \boldsymbol{\epsilon} = 0 \\
\mathbb{E}(y_i) &= \sum_{j=0}^{P-1} X_{i,j}\beta_j && \text{for the each element} \\
&= X_{i,*}\beta_i && \text{where } _* \text{ replace the sum over index } i
\end{aligned}$$

The variance for the element y_i can be found by

$$\begin{aligned}
\mathbb{V}(y_i) &= \mathbb{E}\big[(y_i - \mathbb{E}(y_i))^2\big] \\
&= \mathbb{E}(y_i^2) - (\mathbb{E}(y_i)^2) \\
&= \mathbb{E}((X_{i,*}\beta_i + \epsilon_i)^2) - (X_{i,*}\beta_i)^2 \\
&= \mathbb{E}((X_{i,*}\beta_i)^2 + 2\epsilon_i X_{i,*}\beta_i + \epsilon^2) - (X_{i,*}\beta_i)^2 \\
&= \mathbb{E}((X_{i,*}\beta_i)^2) + \mathbb{E}(2\epsilon_i X_{i,*}\beta_i) + \mathbb{E}(\epsilon^2) - (X_{i,*}\beta_i)^2 \\
&= (X_{i,*}\beta_i)^2 + \mathbb{E}(\epsilon^2) - (X_{i,*}\beta_i)^2 \\
&= \mathbb{E}(\epsilon^2) = \sigma^2
\end{aligned}$$

The expression for the optimal parameter

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

We find the expected value of $\hat{\boldsymbol{\beta}}$

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}) && \text{using that } \boldsymbol{X} \text{ is a non-stochastic variable} \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} && \text{using } \mathbb{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta} \\
&= \boldsymbol{\beta}
\end{aligned}$$

We can find the variance by

$$\begin{aligned}
\mathbb{V}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\big[(\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}}))^2\big] \\
&= \mathbb{E}(\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T) - \mathbb{E}(\hat{\boldsymbol{\beta}})^2 \\
&= \mathbb{E}(((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y})^T) - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}) - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}) - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}
\end{aligned}$$

Knowing the expectation value and the variance of $\hat{\boldsymbol{\beta}}$, we can define a confidence interval for each $\hat{\beta}_j \pm std(\hat{\beta}_j)$ for $j = 1, 2, \ldots, P-1$. The optimal $\hat{\boldsymbol{\beta}}^{Ridge}$ can be derived from MSE, and is defined as

$$\hat{\boldsymbol{\beta}}^{Ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

The expectation value is then

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\beta}}^{Ridge}) &= \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}) \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}) && \text{since } \boldsymbol{X} \text{ and } \lambda\boldsymbol{I} \text{ are non-stochastic variables} \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} && \text{using } \mathbb{E}(\boldsymbol{y}) \text{ from exercise 1}
\end{aligned}$$

For $\lambda = 0$ we have $\mathbb{E}(\hat{\boldsymbol{\beta}}^{OLS})$. The variance

$$\begin{aligned}
\mathbb{V}(\hat{\boldsymbol{\beta}}^{Ridge}) &= \mathbb{E}(\hat{\boldsymbol{\beta}}_R\hat{\boldsymbol{\beta}}_R^T) - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= \mathbb{E}(((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y})(((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}))^T) - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}\boldsymbol{y}^T\boldsymbol{X}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1})^T) - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{X}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1})^T - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1})^T - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{X}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1})^T + (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 - (\mathbb{E}(\hat{\boldsymbol{\beta}}_R))^2 \\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{X}((\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1})^T
\end{aligned}$$

Appendix B: Bias-variance trade-off

From equation $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$ and the assumption $f(\boldsymbol{x}) \approx \tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$ The expectation value of the mean square error is

$$\begin{aligned}
\mathbb{E}((\boldsymbol{y} - \tilde{\boldsymbol{y}})^2) &= \mathbb{E}(\boldsymbol{y}^2) - 2\mathbb{E}(\boldsymbol{y}\tilde{\boldsymbol{y}})\mathbb{E}(\tilde{\boldsymbol{y}}^2) \\
&= \mathbb{E}(f(\boldsymbol{x})^2) + \mathbb{E}(\boldsymbol{\epsilon}^2) - 2f(\boldsymbol{x})\mathbb{E}(\tilde{\boldsymbol{y}}) + \mathbb{V}(\tilde{\boldsymbol{y}}) + \mathbb{E}(\tilde{\boldsymbol{y}})^2 \\
&= f(\boldsymbol{x})^2 - 2f(\boldsymbol{x})\mathbb{E}(\tilde{\boldsymbol{y}}) + \mathbb{E}(\tilde{\boldsymbol{y}})^2 + \mathbb{V}(\tilde{\boldsymbol{y}}) + \mathbb{V}(\boldsymbol{\epsilon}) \\
&= \mathbb{E}((f(\boldsymbol{x}) - \mathbb{E}(\tilde{\boldsymbol{y}}))^2) + \mathbb{V}(\tilde{\boldsymbol{y}}) + \sigma^2
\end{aligned}$$

Appendix C: Plots

Here we have collected some interesting plots from the analysis of the Franke function and the terrain data :)

1. Franke function

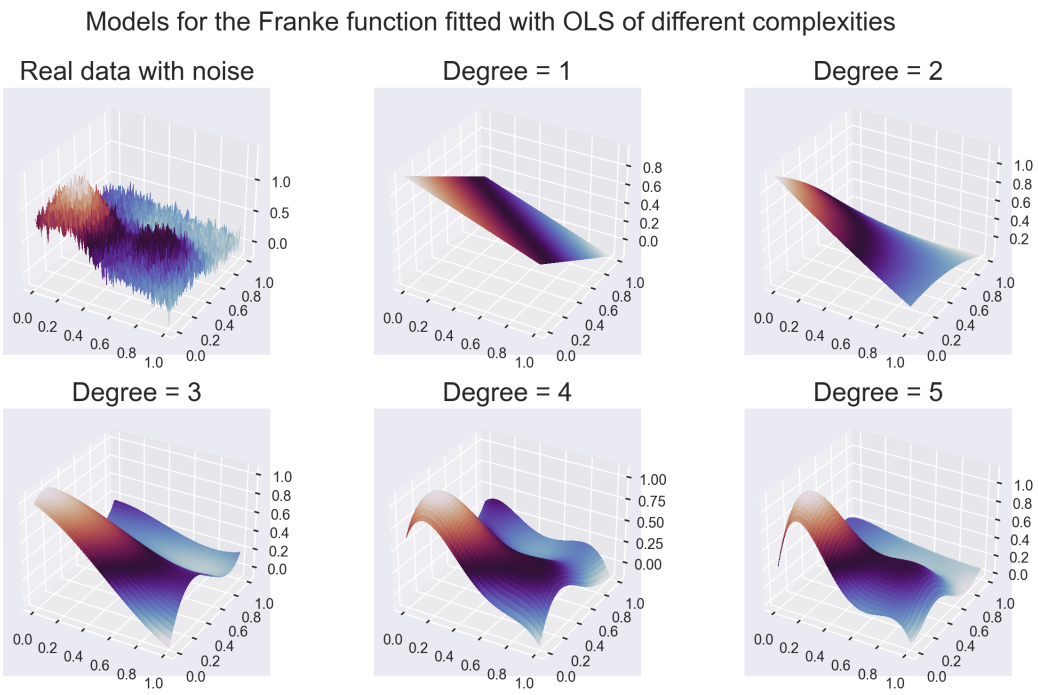

Figure 37. A plot showing how a model of different complexities fit the franke function when OLS regession has been used.

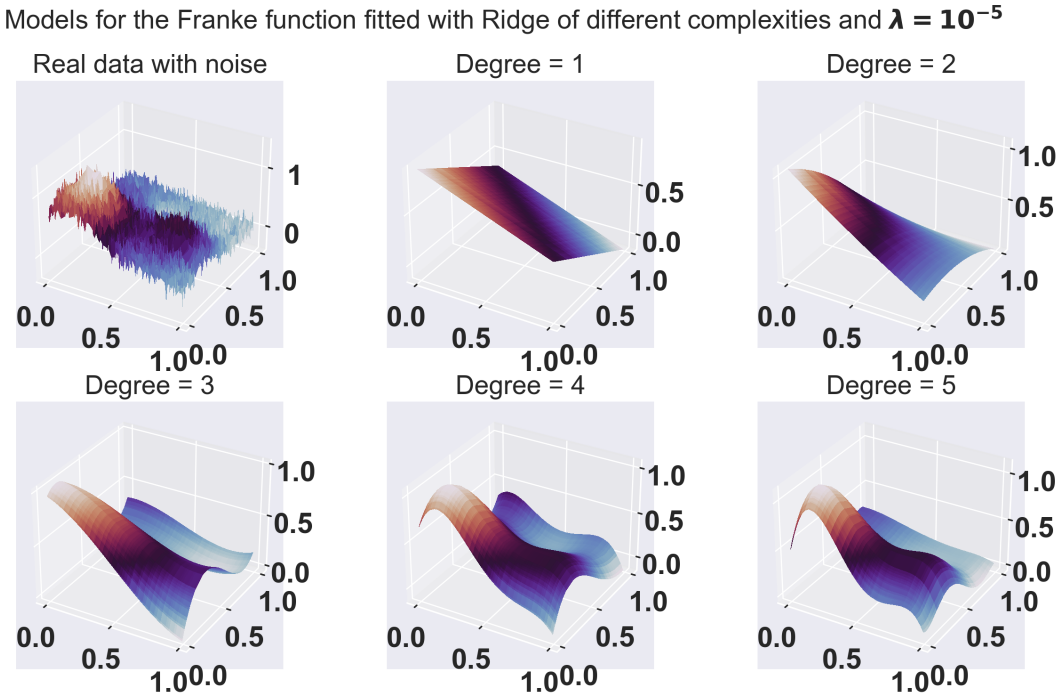

Figure 38.

2. Terrain data

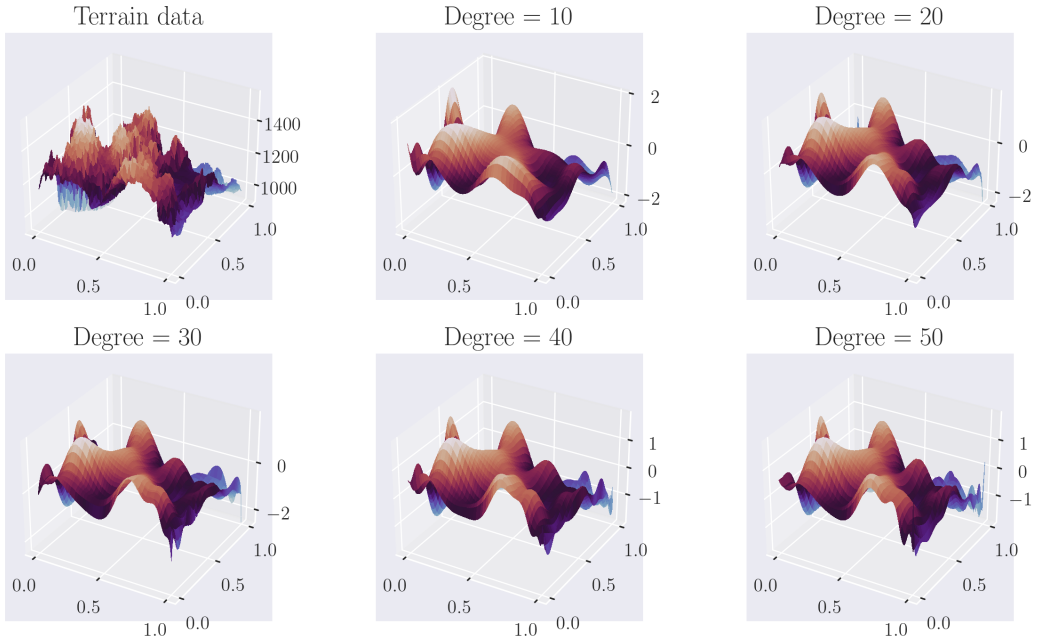

Figure 39. A 3D plot of the terrain data compared to models created with OLS of complexity 10, 20, 30, 40 and 50

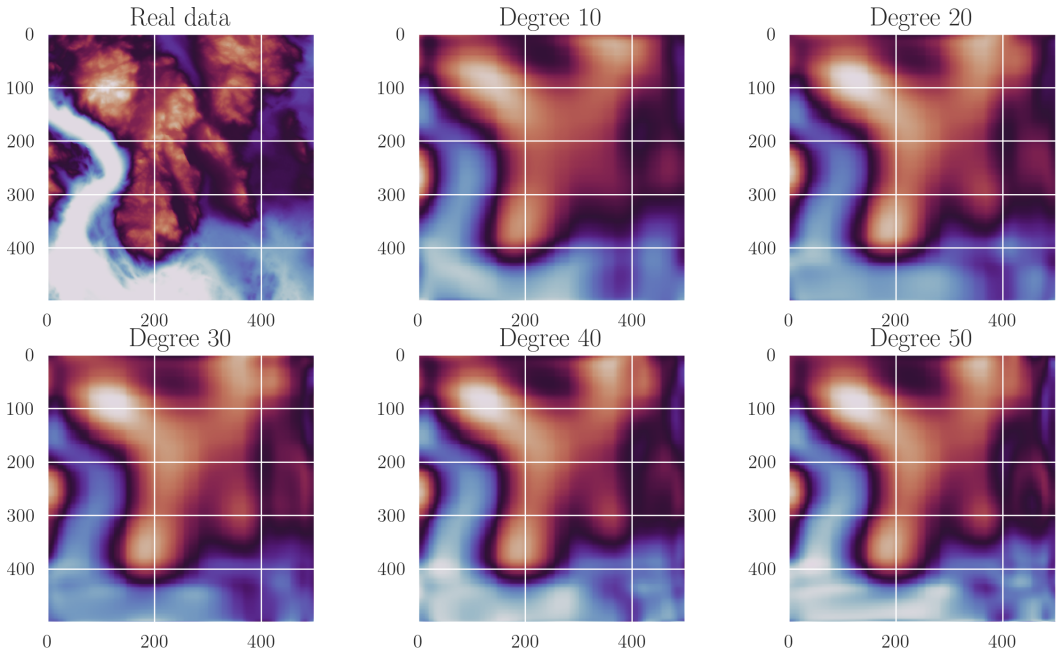

Figure 40. A 2D plot of the terrain data compered to models created with OLS of complexity 10, 20, 30, 40 and 50

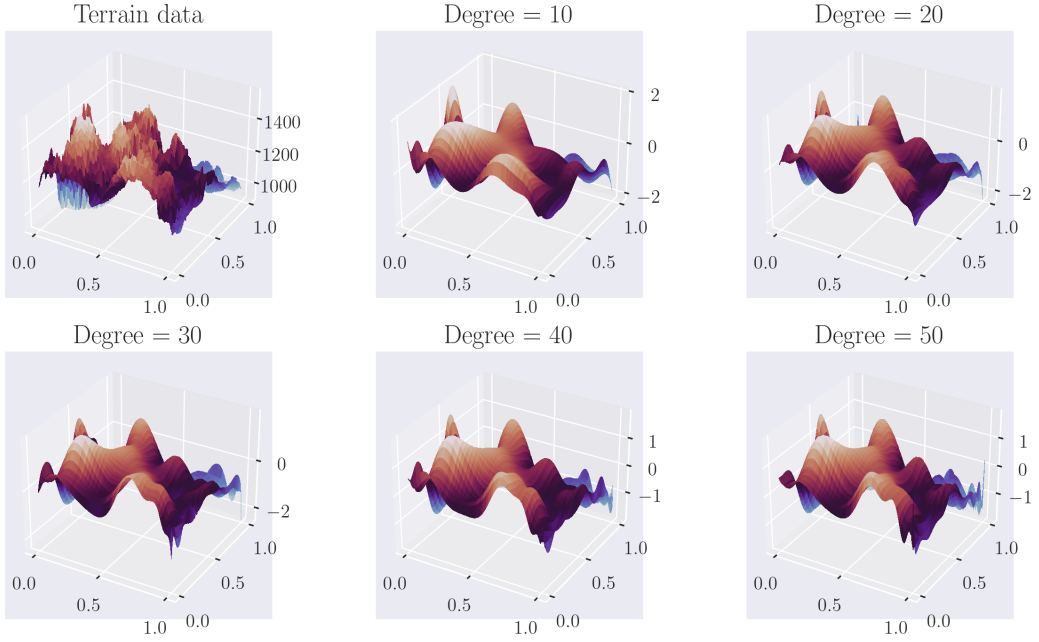

Figure 41. A 3D plot of the terrain data compered to models created with Ridge of complexity 10, 20, 30, 40 and 50 and a λ value of 10^{-5}

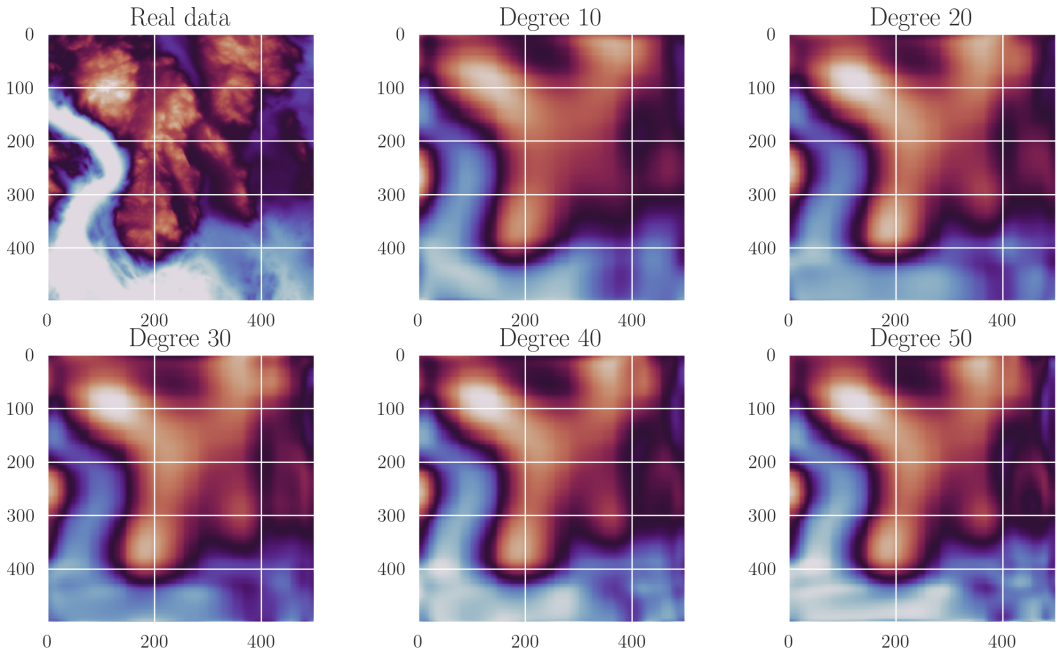

Figure 42. A 2D plot of the terrain data compered to models created with Ridge of complexity 10, 20, 30, 40 and 50 and a λ value of 10^{-5}

REFERENCES

[1] Jason Brownlee. A gentle introduction to the bootstrap method. https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/, 2019.

[2] Trevor Hastie, Robert Tibshiran, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[3] Morten Hjorth-Jensen. Week 34: Introduction to the course, logistics and practicalities. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/week34.html, 2023.

[4] Eberly College of Science. Bootstrapping methods. https://online.stat.psu.edu/stat500/lesson/11/11.2/11.2.1.

[5] John Smith. *Jay L. Devore and Kenneth N. Berk and Matthew A. Carlton*, pages 371–372. Springer Cham, 3 edition, 2021. The Central Limit theorem.

[6] John Smith. *Jay L. Devore and Kenneth N. Berk and Matthew A. Carlton*, pages 376–377. Springer Cham, 3 edition, 2021. The Law of Large Numbers.

[7] Robert Tibshirani Trevor Hastie and Jerome Friedman. *Linear Methods for Regression*, pages 43–94. Springer New York, 2 edition, 2009. Chapter 3.

[8] Wessel N. van Wieringen. Lecture notes on ridge regression. https://browse.arxiv.org/pdf/1509.09169.pdf, 2023.

[9] Hong Wen, Xiyong Wu, Xin Liao, Dong Wang, Kaiyang Huang, and Bernd Wünnemann. Application of machine learning methods for snow avalanche susceptibility mapping in the parlung tsangpo catchment, southeastern qinghai-tibet plateau. *Cold Regions Science and Technology*, 198:103535, 2022.