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 6, 2023)

The aim of this report is to see how different regression method affects the data it is applied to. More concretely, we will look at the three different methods ordinary least squares (OLS), Ridge and LASSO. We will also apply bias variance trade of as well as cross validation on the data sets used to evaluate our models. What we found was that ... regression with parameter ... best fitted the topological data analysis...

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...

I. INTRODUCTION

Machine learning is a powerfull tool usfull in many fields of reasarche. 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[3], potentially leading to life-saving interventions. This methods can extends to addressing concerns related to floods, which has become a hot topic this past month following the storm Hans. It can also help in aiding with spatial planning challenges. which is usefull in big citys all over the world. It is fair to say machine learning possesses immense potential to contribute to the solutions of 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 and see how these method compare to eachother when applied to different data sets. First we are going to use the Franke function to make dummy data to validate if our models works. 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 on real digital terrain data taken from https://earthexplorer.usgs.gov/. To more accuraltly simulate the realism of practical machine learning scenarios, we will impose limitations on our datasets. Additionally, we will employ techniques such as bootstrapping and cross-validation to expand our dataset size and assess their impact on model validation.

II. THEORY

A. Linear regression methods

Linear regression is a foundational statistical modeling technique employed for the prediction of 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, which is mathematically expressed as:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{1}$$

In this equation, $\hat{y}$ denotes the predicted target variable, while $x_1, x_2, \ldots, x_p$ represent the input features. The coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are parameters that correspond to each respective feature. The primary objective of linear regression is to determine these coefficients to establish the optimal linear model that best fits the given data.

Linear regression encompasses several variants, each offering 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) stands as a fundamental technique in regression analysis, where the aim is to create a model that minimize the diffrense from the observed data and the predicted model.

A linear regression model has the following form :

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

Where β is the coefficients and X the design matrix. It is importent to note that thise method assume that either the function is linear or approxematly linear.

2. Ordinary least squares (OLS)

Ordinary Least Squares (OLS) stands as a fundamental technique in regression analysis, with the primary objective of creating a model that minimizes the discrepancy between observed data and the predicted model. This is achieved by estimating the coefficients in a linear regression model, as defined in Equation (2) , while minimizing the Mean Squared Error (MSE) To minimize the MSE, we calculate the optimal β values. This involves taking the derivative of the cost function, as shown in Equation (3), and setting it to zero.

$$C(\beta) = \frac{1}{n} \left\{ (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \right\} \tag{3}$$

By solving $\frac{\partial C(\beta)}{\partial \beta} = 0$, we determine that for OLS, the optimal β values can be obtained by solving Equation (4).

$$\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{4}$$

Additionally, OLS possesses other properties that are relevant to this project, such as the expectation value and variance. These properties come into play when applying the bias-variance trade-off to evaluate our models, and they can be expressed as follows:

$$\mathbb{E}_{OLS}(y_i) = \mathbf{X}i, *\beta \tag{5}$$
$$\mathbb{E}_{OLS}(\beta) = \beta \tag{6}$$
$$\mathbb{V}_{OLS}(y_i) = \sigma^2 \tag{7}$$
$$\mathbb{V}_{OLS}(\hat{\beta}) = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1} \tag{8}$$

For a comprehensive step-by-step derivation of these expressions, please refer to Appendix A. OLS, as a regression method, relies on specific assumptions about the data it is applied to. If these assumptions are not met by the dataset, the model's accuracy may be compromised.

Write about positiv and negativ things about OLS

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

3. Ridge

Ridge regression closely resembles OLS,but it have the addition of a term in the cost function, as depicted in the equation (9).

$$C(X, \beta) = \left\{ (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \right\} + \lambda \beta^T \beta \tag{9}$$

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 optain the equation for the optimal β:

$$\beta = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \tag{10}$$

This extra term $\lambda \mathbf{I}$ helps controling the size of the coefficents, if λ is small, then ..

For the Bias variance trade off we will also need the following properties:

$$\mathbb{E}(\beta_{Ridge}) = (X^T X + \lambda I)^{-1} (X^T X) \beta \tag{11}$$
$$\mathbb{V}(\beta_{Ridge}) = \sigma^2 [X^T X + \lambda I]^{-1} X^T X \{ [X^T X + \lambda I]^{-1} \}^T \tag{12}$$

Calculations for both the optimal β and variance can be found in the appendix

4. LASSO

Lasso, short for Least Absolute Shrinkage and Selection Operator, 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. (Kilde 1AM)
The cost function for Lasso is shown under in equation (13).

$$C(X, \beta)_{\text{Lasso}} = \left\{ (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \right\} + \lambda ||\boldsymbol{\beta}||_1 \tag{13}$$

where $||\boldsymbol{\beta}||_1 = \sum_i |\boldsymbol{\beta_i}|$ which is found by taking the derivative with respect to $\boldsymbol{\beta}$ (Kilde 1AM),

$$\mathbf{X}^T \mathbf{X} \boldsymbol{\beta} + \lambda sgn(\boldsymbol{\beta}) = 2 \mathbf{X}^T \boldsymbol{y} \tag{14}$$

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.(Kilde 2AM)

B. MSE

C. Resampling techniques

The main restriction in machine learning is the amount of data points available to create the model out of. It may be the case where one have done a costfull and time consuming experiments and are left with a small number of data. It is therfull extremely useful to have methods where one can reuse the data multiple times thereby creating a relatively large dataset from the small number of datapoints. In this report we are going to use two different methods, the first is called bootstrap and the second one is cross validation.

1. Bootstrap

The bootstrap method is a resampling procedure that uses data from one sample to generate a sampling distribution by repeatedly taking random samples from the known sample, with replacement This means that if we have a data set D with n data points. The elements in this data set can be representatet in the following way:

$$D = d_1, d_2, d_3, d_4, \ldots, d_n \tag{15}$$

Then by apllying the bootstarp method on this data set one possible output D^* can be:

$$D^* = d_3, d_n, d_4, d_4, \ldots, d_2 \tag{16}$$

From this example we see that one observation can appear multiple times in the new dataset. We can take this method a step further and create "new" data points by extracting multible data points from the dataset and take the mean of all thise values:

$$d_{new} = \frac{1}{k}(d_1, \ldots d_k) \tag{17}$$

This gives us a method of producing lots of "new" dataset from limited data points to train our model with. For each of this data-sets the mean and standard deviation can be calculated to evaluate the model statistically. [1]

One huge advatange of using the boostrap method is that the data can be split in to test and train before shuffeling the data, this means that the test data can be kept entirely separate from the creation of the model. When we den test the model it will be on a dataset that has nothing to do with crating the model and will therfore show how god the model represent real data. **Write something about large numbers law, also disadvantages and advantage**

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 dataset. Unlike the bootstrap method, which samples with replacement, CV partitions the original dataset 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 dataset D with n data points represented as:

$$D = \{d_1, d_2, d_3, \ldots, d_n\} \tag{18}$$

For $k = 5$, an example partition might be:

$$\text{Fold } 1 = \{d_1, d_2, \ldots, d_{\frac{n}{5}}\} \tag{19}$$

$$\text{Fold } 2 = \{d_{\frac{n}{5}+1}, d_{\frac{n}{5}+2}, \ldots, d_{2\times\frac{n}{5}}\} \tag{20}$$

$$\vdots \tag{21}$$

$$\text{Fold } 5 = \{d_{4\times\frac{n}{5}+1}, d_{4\times\frac{n}{5}+2}, \ldots, d_n\} \tag{22}$$

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 dataset for both training and testing, which can be very helpfull 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 datasets. 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.

D. Bias-variance trade-off

The Bias-varinace trade-off is a measuerment on how accuratly a model fit the real data.

III. METHOD

In the first part of this project a function called Franke function was used as the data analysed. The Franke func-

tion is given by the following equation:

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

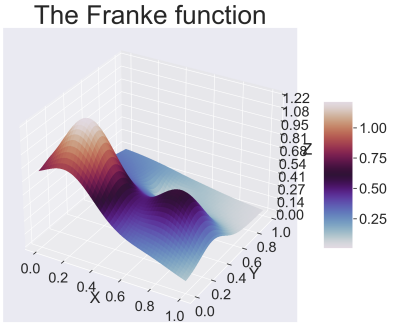

Figure 1. A plot of the Franke function

The amount of data created of the franke function was a $20x20$ matrix where 20% of the data set was set aside as the testing dataset while the ramaining 80% was used for training. The noise was reduced from $\mathcal{N}(0,1)$ as proposed in the project description, to $\prime,\prime.\infty$ for better results. First this data was fitted with the OLS method were varying degrees of polynomials was used to create the design matrix. Since the design matrix in this case was noninvertible, singular value decomposition was used to create 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 datasets. The different coefficients, or so called 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 ...
Next the bootstrap method was implementet with .... iterations to se how this affected the result of the OLS regression analysis. This method was implemented after the dataset was split into a training and testing set, so that the testing dataset didn't have anything to do with the creation of the diffrent β values.
The last thing done with OLS on the franke function was cross validation..

Need to rewrite the method section about cross validation 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 dataset 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 λ value in a logarithmically spaced array of λ values, denoted `lambdas`, the Ridge regression model was trained and validated k times - once per fold. Specifically, for every tuple (train_indices, test_indices) produced by `k_fold(data, k)`, the data was divided into training and test sets $(x_{\text{train}}, y_{\text{train}}, x_{\text{test}}, y_{\text{test}})$. Subsequently, the `design_matrix` function generated polynomial feature matrices X_{train} and X_{test} from the x and y values, using a specified polynomial degree.

The Ridge model, instantiated with the current λ, was then fitted with X_{train} and y_{train}, and predictions y_{pred} were made using X_{test}. The Mean Squared Error (MSE) between the predictions and actual test values y_{test} 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.

Next Ridge and LASSO regression was used on the Franke function, to see if these methods have a better fit than what was obtained with OLS. Diffrent values for λ was used to obtain the best fit as possible for each polynomial degree. For Ridge equation (10) was used to calculate the cofficents, Meanwhile, for LASSO regression, the calculations were performed using scikit-learn's LASSO regression function from its linear regression package.

In the last part of this project real terrain data was analysed. Due to the massiv size of the terrain data only the first $500 \cdot 500$ matrix as shown in figure (2) was used to avoid ram problems. The OLS regression was applyed

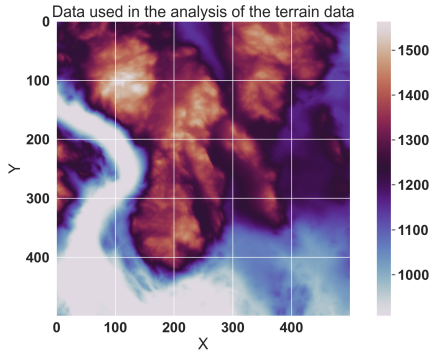

Figure 2. A plot of the data matrix used in the analysis of the terrain data in the second part of project 1.

to creat a modell of the dataset

IV. RESULTS

A. Result for Franke function

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 dataset conisted of $20 \cdot 20$ data points. The results for the MSE and R2 score is shown in figure (3)

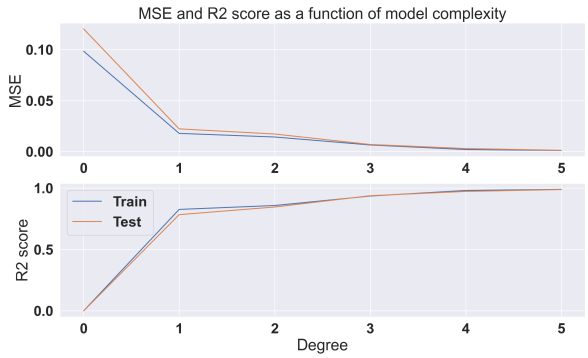

Figure 3. Plot showing the MSE and R2 score for the franke function without noise. The regression method used was OLS

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 includet in to the data set.

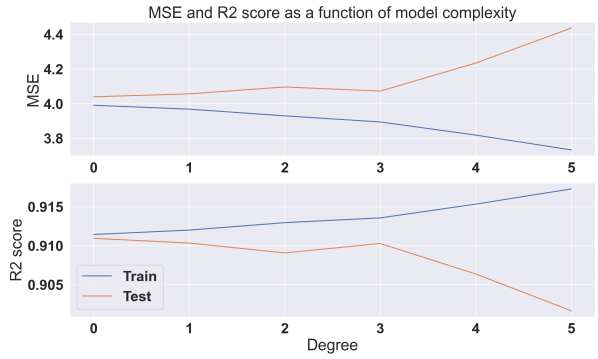

Figure 4. Plot showing the MSE and R2 score for the franke function with noise $\mathcal{N}(0, 0.1)$. The regression method used was OLS

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

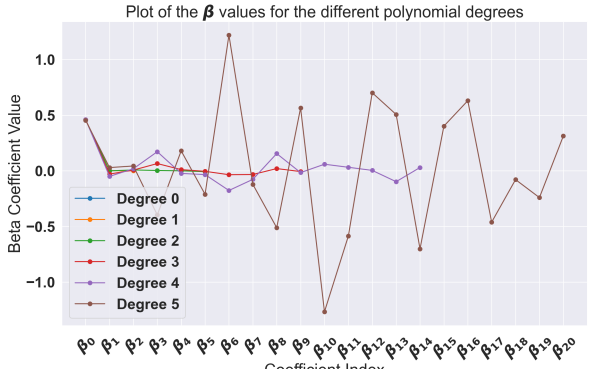

Figure 5. A plott showing the β values for OLS for different orders of polynimials

For Ridge Regression on the franke function have we plotted a heatmap to show how the MSE changes with diffrent λ values and complexities. A plot of the heatmap for the training data is shown in figure (6) , and the heatmap for the test data is shown in figure (7).

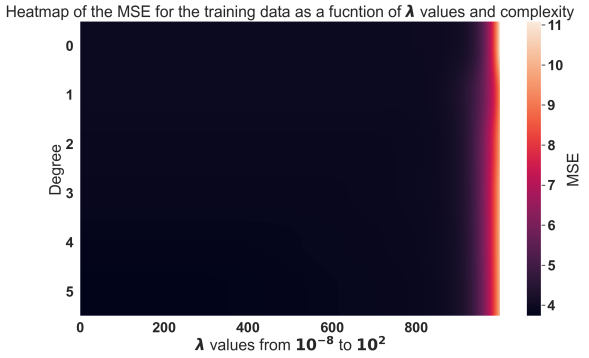

Figure 6. A heatmap of the MSE for the training data, for different λ values and complexities. The λ values goes from 10^{-8} to 10^2

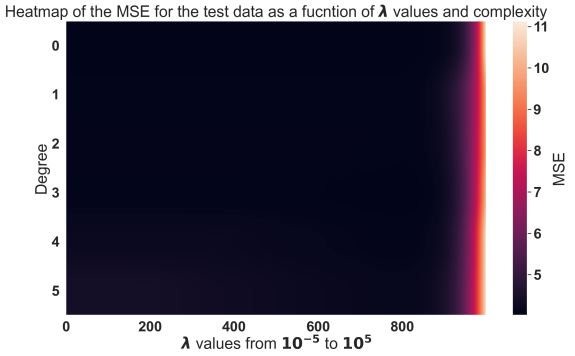

Figure 7. A heatmap of the MSE for the testing data, for different λ values and complexities. The λ values goes from 10^{-5} to 10^5

Lastly for Ridge regression we see what happens when λ is put to zero.

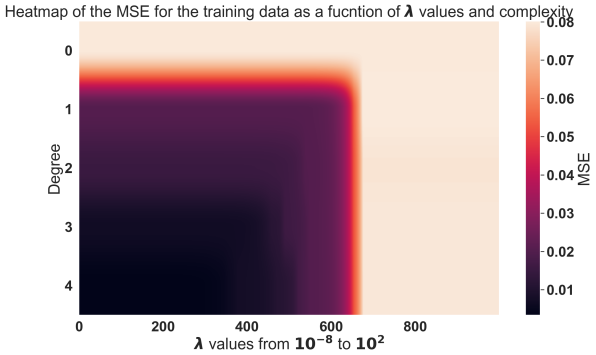

Figure 8. A heatmap of the MSE for the training data, for different λ values and complexities. The λ values goes from 10^{-8} to 10^2

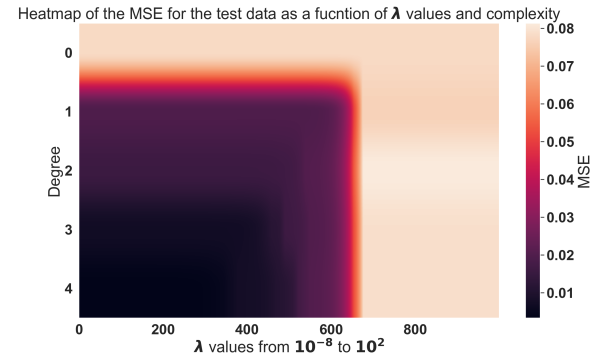

Figure 9. A heatmap of the MSE for the testing data, for different λ values and complexities. The λ values goes from 10^{-5} to 10^5

B. Results for terrain data

V. DISCUSSION

It was stated in the project description that we shoud use a nois that was given by the normal distribution $\mathcal{N}(0,1)$. What we found was that since the Franke Function when plotten for x and y in the range between 0 and 1 the maximal value of the Franke function becomes around 1.4. When plottet with noise that had values between 0 and 1 the plot of the franke function becomes unrecognizable as shown in figure (13).

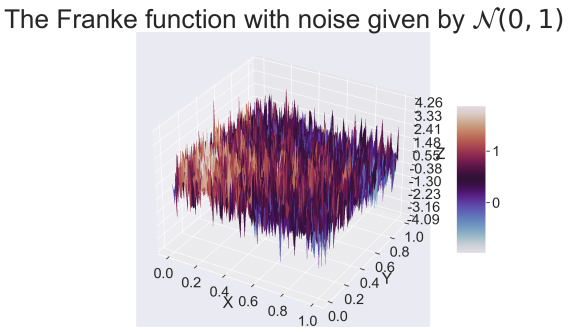

Figure 13. Plot showing the Frankece function with noise given by the normal distrubution $\mathcal{N}(0,1)$

But if instead the normal distribution $\mathcal{N}(0,0.1)$ is used, the dataset becomes much easier to work with, and the results becomes much prettier (as shown in figure (14)) since noise has a tendency of ruining everything.

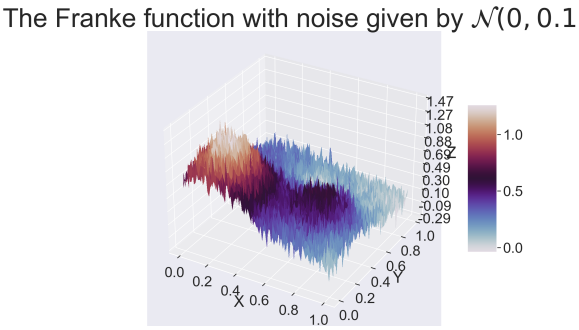

Figure 14. Plot showing the Frankece function with noise given by the normal distrubution $\mathcal{N}(0,0.1)$

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.

The Franke function

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 dataset 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 posibility 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 det test set we see a sudden increas after fitting with a fourth order polynomial. This is due to the amount of datapoints in the diffrent datasets, since the test dataset only have 20% of the original data while the training set consist of the ramaining 80%. If we increas the number of data points the predictet 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 increas in complexity, while the R2 score gets closer and closer to 1. But when we introduce noise whe see from figure (4) that for our test data the MSE acctually increase with higher complexity.

The dataset containing the test data consists of 20% of the original dataset. This implies that the number of points in the test data is significantly smaller than that in the training data. We anticipate that by increasing the number of data points, the MSE for the model created using the test data will converge towards that of the training model.

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 overfit area.

We expect that if we set $\lambda = 0$ in our Ridge regression then the result will be the same as for OLS. From figur .. and ... we se that when lambda is set to zero then the model becoms equal to that for OLS, explenation...

Real terrain data

VI. CONCLUSION

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.

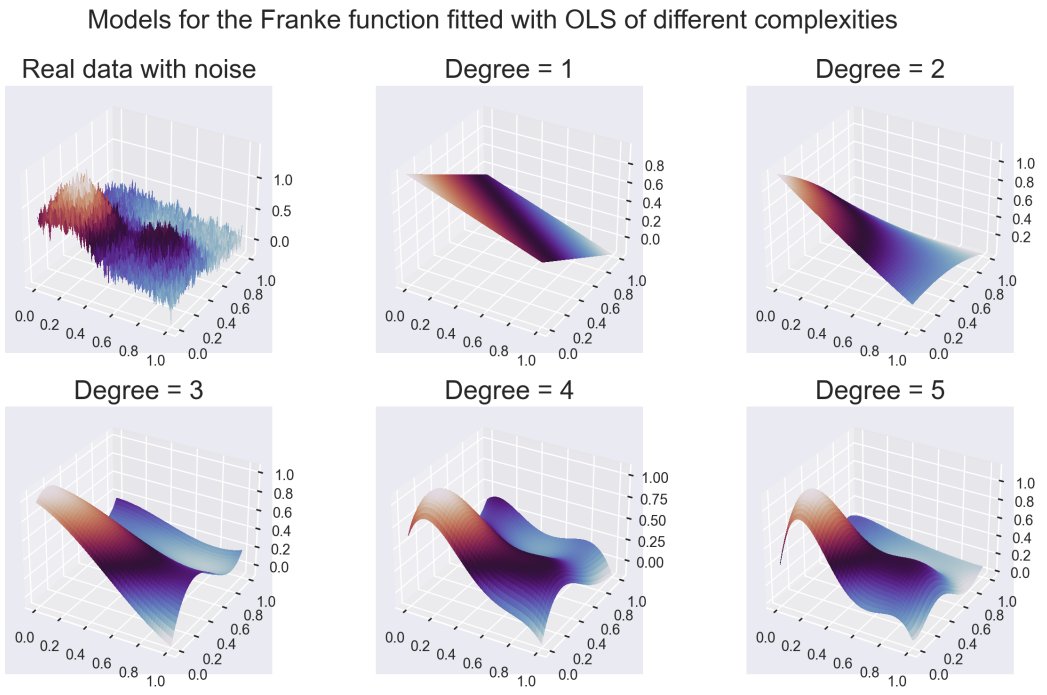

Figure 10. A plot showing how model with different complexities fit the franke function when OLS regession has been used.

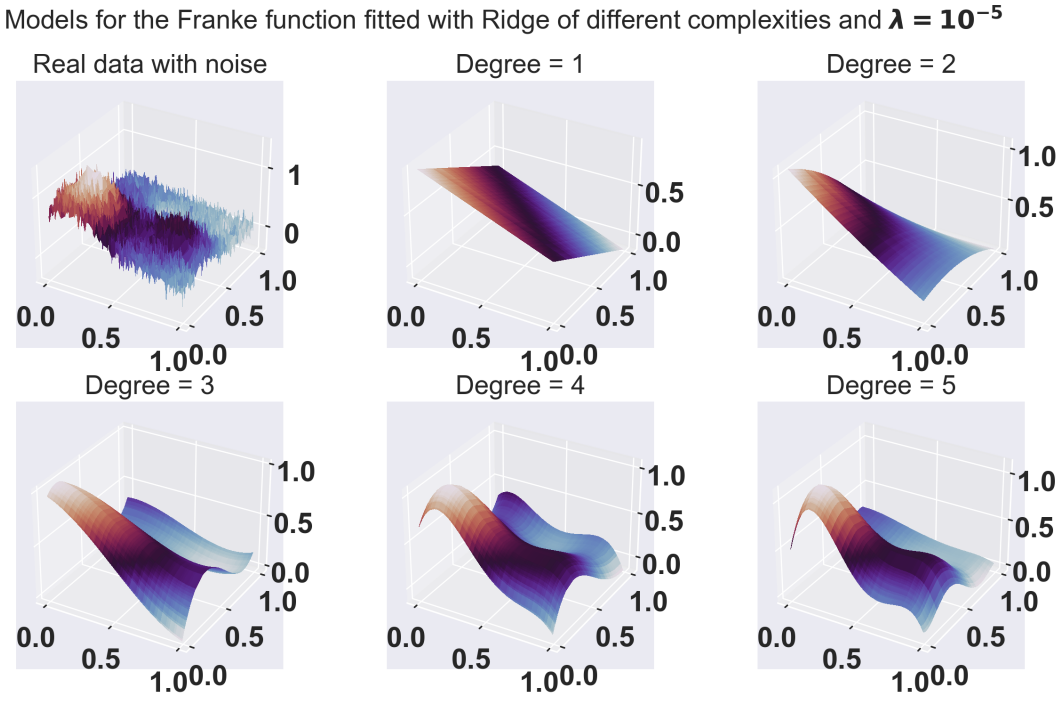

Figure 11.

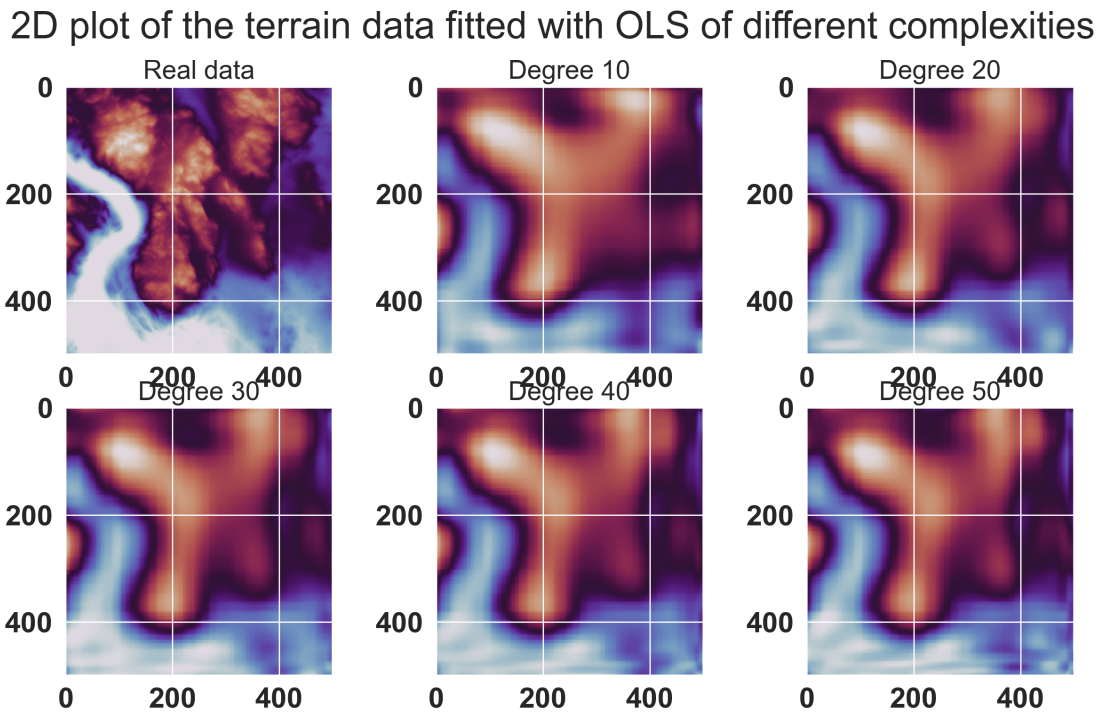
2D plot of the terrain data fitted with OLS of different complexities
Real data
Degree 10
Degree 20
Degree 30
Degree 40
Degree 50
0
200
400

Figure 12.

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 Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[3] 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.

Appendix A: Mean values and variances calculations

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 intervall 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