FYS-STK4155: Project 1

Camilla Dalby Borger

October 2023

Abstract

This scientific report investigates various regression methods, including Ordinary Least Squares (OLS), Ridge, and Lasso regression, applied to the Franke function. The primary objective is to analyze and compare the Mean Squared Error (MSE) and the R2-score for these regression techniques. In the case of Ridge and Lasso regression, an extensive examination is conducted by varying the regularization parameter, λ, to assess its impact on the MSE and R2-score.

Furthermore, the study explores the concept of the bias-variance trade-off in continuous prediction tasks, focusing exclusively on Ordinary Least Squares regression for simplicity. The Bootstrap resampling technique is employed to analyze the MSE as a function of model complexity, the number of data points, and the division between training and test data.

Additionally, the research incorporates the Cross-Validation method, comparing its MSE results with those obtained from Bootstrap resampling. The investigation varies the number of folds in Cross-Validation (ranging from 5 to 10) and compares our own Cross-Validation code with the one provided by Scikit-Learn. Ridge and Lasso regression methods are also evaluated within the context of Cross-Validation.

The culmination of this analysis involves applying these regression methods to real-world terrain data, extending the findings from synthetic data (Franke function) to practical applications.

Introduction

Regression analysis plays a pivotal role in the field of statistics and data science, serving as a cornerstone for making predictions and uncovering underlying relationships within datasets. It offers a powerful toolset for modeling and understanding complex phenomena, making it indispensable in various scientific and practical domains. In this report, we delve into the realm of regression analysis, specifically focusing on the application of three prominent techniques: Ordinary Least Squares (OLS), Ridge, and Lasso regression.

Understanding and comparing these regression methods is crucial for both researchers and practitioners. It empowers us with the ability to select the most suitable model for our specific data and objectives, thereby enhancing the quality of our predictions. In a world inundated with data-driven decision-making, the ability to navigate the intricacies of these methods equips us with valuable skills applicable to a wide range of domains, from finance to healthcare, from environmental science to engineering.

In the course of this project, we have undertaken a comprehensive exploration of these regression methods. We began by applying OLS, Ridge, and Lasso regression to the well-known Franke function, providing a foundation for our comparative analysis. Our primary aim was to assess and contrast the performance of these techniques by evaluating two essential metrics: the Mean Squared Error (MSE) and the R2-score. For Ridge and Lasso regression, we con-

ducted an in-depth investigation into the effect of the regularization parameter, λ, on the model's predictive accuracy.

Furthermore, we addressed the critical concept of the bias-variance trade-off in continuous prediction tasks, employing the Bootstrap resampling technique to delve into the nuances of model complexity, data volume, and the impact of data division into training and test sets. To further enhance our understanding, we examined these concepts within the context of Ordinary Least Squares regression.

Our exploration extended to the incorporation of the Cross-Validation method, enabling us to contrast its performance against that of Bootstrap resampling. We systematically varied the number of folds in Cross-Validation, ranging from 5 to 10, and conducted a comparative analysis between custom-developed Cross-Validation code and the implementation provided by Scikit-Learn. Additionally, we expanded our analysis to include Ridge and Lasso regression within the framework of Cross-Validation.

This report is organized into several sections to facilitate a systematic presentation of our findings and analyses. Following this introduction, we delve into the theoretical underpinnings of the regression methods in the "Methods" section. The "Code" section outlines our experimental setup, including data sources, parameter choices, and resampling techniques.

In the subsequent sections, we present our results and analyses in a structured manner, focusing first on the comparative evaluation of OLS, Ridge, and Lasso regression using synthetic data (Franke function). We then explore the bias-variance trade-off and the implications of Bootstrap resampling. Following that, we investigate the application of the Cross-Validation method, providing insights into its performance and its comparison to Bootstrap resampling, especially concerning Ridge and Lasso regression.

Lastly, we apply the knowledge gained from synthetic data to a real-world scenario, specifically, the analysis of terrain data, showcasing the practical implications of our findings. The report concludes with a summary of key takeaways, highlighting the significance of regression analysis in various domains.

Methods

In this section, we describe the methods and techniques used in our study to build and evaluate regression models. The primary objective is to predict a target variable based on a set of features. The following section is based on the lecture notes [1].

Data Preprocessing

Data Splitting

It is normal in essentially all Machine Learning studies to split the data in a training set and a test set (sometimes also an additional validation set). There is no explicit recipe for how much data should be included as training data and say test data. In this project We divided the data sets into two parts: a training set (80% of the data) and a testing set (20% of the data) to ensure that the model's performance can be evaluated effectively.

Data Centering and Scaling

Before fitting the regression models, we could apply centering and scaling to the input features to ensure that all variables have a mean of 0 and a standard deviation of 1. This preprocessing step aids in numerical stability and model convergence. We should have done this for the Terrain data, but not for the Franke function.

Regression Models

Regression models are fundamental tools in statistics and machine learning used to explore relationships between one or more independent variables (predictors) and a dependent variable (target). In its simplest form, linear regression models the relationship between the predictors ($\mathbf{x}^T = [x_0, x_1, x_2, ..., x_{n-1}]$) and the target variable ($\mathbf{y}^T = [y_0, y_1, y_2, ..., y_{n-1}]$) using a linear equation:

$$\mathbf{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \boldsymbol{\epsilon}$$

Here, β_0 represents the intercept, β_i are the coefficients for the respective predictors, x_i, and ϵ is the error term. The goal of regression is to estimate the coefficients (β_i) that minimize the sum of squared errors (ϵ), often done using methods such as Ordinary Least Squares (OLS). More advanced regression methods like Ridge and Lasso introduce regularization terms to prevent overfitting. These models offer a flexible framework for understanding and predicting complex relationships in various domains, from economics to machine learning. The choice of the appropriate regression model depends on the nature of the data and the trade-off between model complexity and interpretability. We employed the following regression techniques to model our data.

Ordinary Least Squares (OLS)

For OLS, or Ordinary Least Squares, is a classical and widely-used method in linear regression. It aims to estimate the coefficients of a linear model that minimizes the sum of squared differences between the observed values and the values predicted by the model. Lets assume assume that the output data can be representsed by a continuous function f through

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

We approximate the unknown function with another continuous function

$$\tilde{\mathbf{y}}(\mathbf{x}) = \mathbf{X}\boldsymbol{\beta}$$

, and in order to find the optimal parameters β_i we define a function which gives a measure of the spread between the values y_i (which represent the output values we want to reproduce) and the parametrized values $\tilde{y_i}$, namely the so-called cost/loss function. We use the mean squared error.

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2 = \frac{1}{n}\left\{(\mathbf{y} - \tilde{\mathbf{y}})^T(\mathbf{y} - \tilde{\mathbf{y}})\right\},$$

or using the matrix $\mathbf{X}$ and in a more compact matrix-vector naation as

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

In order to find the parameters β_i we will the minimize the spread of $C(\boldsymbol{\beta})$, that is we are going to solve the problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\left\{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right\}.$$

In practical terms it means we will require

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

We can rewrite this as

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{y},$$

and if the matrix $\mathbf{X}^T\mathbf{X}$ is invertible we have the soltuion

$$\boldsymbol{\beta}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

The OLS estimator for the coefficients provides a straightforward and interpretable way to model linear relationships between variables, making it a foundational technique in regression analysis.

Ridge Regression

Ridge regression is a regularized linear regression method that adds an L2 penalty term to the ordinary least squares objective function, which helps prevent overfitting. Lets us remind ourselves about the expression for the standard Mean Squared Error (MSE) which we used to define our cost function and the equations for the ordinary least squares (OLS) method, that is our optimization problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}} \frac{1}{n}\left\{(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right\}.$$

Or we can state it as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}} \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 = \frac{1}{n}\,||\mathbf{y}-\mathbf{X}\boldsymbol{\beta}||_2^2,$$

where we have used the definition of a norm-2 vector, that is

$$||\mathbf{x}||_2 = \sqrt{\sum_i x_i^2}.$$

By minimizing the above equation with respect to the parameters $\boldsymbol{\beta}$ we could then obtain an analytical expression for the parameters $\boldsymbol{\beta}$. We can add a regularization parameter λ by defining a new cost function to be optimized, that is

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}} \frac{1}{n}\,||\mathbf{y}-\mathbf{X}\boldsymbol{\beta}||_2^2+\lambda||\boldsymbol{\beta}||_2^2,$$

which is the Ridge regression minimization problem where we require that $||\boldsymbol{\beta}||_2^2 \leq t$, where t is a finite number larger than zero. Using the matrix-vector expression for Ridge regression and dropping the parameter $1/n$ in front of the standard means squared error equation, we have

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

, and taking the derivatives with respect to $\boldsymbol{\beta}$ we obtain then a slightly modified matrix inversion problem which for finite values of λ does not suffer from singularity problems. We obtain the optimal parameters

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

with $\mathbf{I}$ being a $p\times p$ identity matrix with the constraint that

$$\sum_{i=0}^{p-1}\beta_i^2 \leq t,$$

with t a finite positive number.

We see that Ridge regression is nothing but the standard OLS with a modified diagonal added to $\mathbf{X}^T\mathbf{X}$. The consequences on the bias-variance tradeoff are rather interesting. We will see that for specific values of λ, we may even reduce the variance of the optimal parameters $\boldsymbol{\beta}$.

Lasso Regression

Lasso regression, similar to ridge regression, is a regularized linear regression technique. It adds an L1 penalty term to the objective function, encouraging sparsity in the model by setting some coefficients to zero.

Using the matrix-vector expression for Lasso regression, we have the following cost function

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}\left\{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right\} + \lambda||\boldsymbol{\beta}||_1.$$

Taking the derivative with respect to $\boldsymbol{\beta}$ and recalling that the derivative of the absolute value is

$$\frac{\partial|\beta|}{\partial|\beta|} = sgn(\beta) = \begin{cases} 1 & \beta > 0 \\ -1 & \beta < 0. \end{cases}$$

We can then take the derivative of the cost function

$$\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\frac{2}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda sgn(\boldsymbol{\beta}) = 0,$$

and reordering we have

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

This equation does not lead to a nice analytical equation as in Ridge regression or OLS. Lets look at the cost function again, now not using the matrix-vector expression, then we have:

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{p-1}(y_i - \beta_i)^2 + \lambda\sum_{i=0}^{p-1}|\beta_i| = \sum_{i=0}^{p-1}(y_i - \beta_i)^2 + \lambda\sum_{i=0}^{p-1}\sqrt{\beta_i^2},$$

and minimizing we have that

$$-2\sum_{i=0}^{p-1}(y_i - \beta_i) + \lambda\sum_{i=0}^{p-1}\frac{(\beta_i)}{|\beta_i|} = 0,$$

which leads to

$$\hat{\beta_i}^{Lasso} = \begin{cases} y_i - \frac{\lambda}{2} & \text{if} \quad y_i > \frac{\lambda}{2} \\ y_i + \frac{\lambda}{2} & \text{if} \quad y_i < -\frac{\lambda}{2} \\ 0 & \text{if} \quad |y_i| \leq \frac{\lambda}{2}. \end{cases}$$

Plotting these results shows clearly that Lasso regression supresses (sets to zero) values of β_i for a specific values of λ.

Model evaluation

Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a commonly used metric to evaluate the performance of regression models. It quantifies the average squared difference between the actual target values (y_i) and the predicted values ($\tilde{y}_i$) produced by the model. MSE is calculated as:

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

where n is the number of data points. Lower MSE values indicate a better fit of the model to the data.

$\mathbf{R^2}$ Score

The R-squared (R^2) score, on the other hand, provides a measure of how well the model explains the variability in the data. It is a value between 0 and 1, where a higher R^2 score indicates a better fit. R^2 is calculated as:

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

where we have defined the mean value of $\boldsymbol{y}$ as

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

R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. A perfect model has an R^2 score of 1, while an R^2 score of 0 indicates that the model does not explain any variance in the data.

Bias-Variance Tradeoff

Lets assume that the true data is generated from a noisy model

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

Here ϵ is normally distributed with mean zero and standard deviation σ^2. In our derivation of the ordinary least squares method we defined then an approximation to the function f in terms of the parameters $\boldsymbol{\beta}$ and the design matrix $\boldsymbol{X}$ which embody our model, that is $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$. The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the mean squared error via the so-called cost function

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

Here the expected value $\mathbb{E}$ is the sample value.

W can show that you can rewrite this in terms of a term which contains the variance of the model itself (the so-called variance term), a term which measures the deviation from the true data and the mean value of the model (the bias term) and finally the variance of the noise. That is we can show that

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

Using the information we have from above, we can write.

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

Solving the terms one by one, using that $\boldsymbol{\epsilon} \sim N(0,\sigma^2)$:

$$\begin{aligned}\mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon})^2] &= \mathbb{E}[\boldsymbol{f}^2] + 2\mathbb{E}[\boldsymbol{f}\boldsymbol{\epsilon}] + \mathbb{E}[\boldsymbol{\epsilon}^2] \\ &= \boldsymbol{f}^2 + \sigma^2\end{aligned}$$

$$\begin{aligned}\mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon})\tilde{\boldsymbol{y}}] &= \mathbb{E}[\boldsymbol{f}\tilde{\boldsymbol{y}}] + \mathbb{E}[\boldsymbol{\epsilon}\tilde{\boldsymbol{y}}] \\ &= \boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}] + \mathbb{E}[\tilde{\boldsymbol{y}}]\mathbb{E}[\boldsymbol{\epsilon}] \\ &= \boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}]\end{aligned}$$

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

Then we can rewrite the above expression

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \boldsymbol{f}^2 + \sigma^2 - 2\boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}] + \boldsymbol{Var}[\tilde{\boldsymbol{y}}] + (\mathbb{E}[\tilde{\boldsymbol{y}}])^2 \\ &= \mathbb{E}\left[(\boldsymbol{y}-\mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + \text{Var}\left[\tilde{\boldsymbol{y}}\right] + \sigma^2 \\ &= (\text{Bias}[\tilde{y}])^2 + \text{var}[\tilde{y}] + \sigma^2\end{aligned}$$

with

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

and

$$\text{var}[\tilde{f}] = \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2.$$

The three terms represent the square of the bias of the learning method, which can be thought of as the error caused by the simplifying assumptions built into the method. The second term represents the variance of the chosen model and finally the last terms is variance of the error $\boldsymbol{\epsilon}$. The variance refers to the amount by which our model would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different estimate. But

ideally the estimate for our model should not vary too much between training sets. However, if a method has high variance then small changes in the training data can result in large changes in the model. In general, more flexible statistical methods have higher variance.

The bias-variance tradeoff summarizes the fundamental tension in machine learning, particularly supervised learning, between the complexity of the model and the amount of training data needed to train it. Since data is often limited, in practice it is often useful to use a less-complex model with higher bias, that is a model whose asymptotic performance is worse than another model because it is easier to train and less sensitive to sampling noise arising from having a finite-sized training dataset (smaller variance).

Understanding the bias-variance tradeoff is essential in model selection. High-bias models tend to underfit the data, while high-variance models tend to overfit. We will analyze this tradeoff during the evaluation of our models.

Resampling Techniques

Resampling methods involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. To assess the performance and generalizability of our models, we employed two resampling techniques:

Bootstrapping

The independent bootstrap works like this:

1. Draw with replacement n numbers for the observed variables $\mathbf{x} = (x_1, x_2, ..., x_n)$.
2. Define a vector $\mathbf{x}^*$ containing the values which were drawn from $\mathbf{x}$.
3. Using the vector $\mathbf{x}^*$ compute $\hat{\beta}^*$ by evaluating $\hat{\beta}$ under the observations $\mathbf{x}^*$.
4. Repeat this process k times.

We used bootstrapping to generate multiple resampled datasets, allowing us to estimate the model's stability and variability in predictions.

Cross-Validation

Cross-validation is a k-fold technique that partitions the data into subsets for training and testing. This approach helps evaluate the model's performance on multiple validation sets and reduces the risk of overfitting.

For the various values of k

1. Shuffle the dataset randomly.
2. Split the dataset into k groups.

3. For each unique group:
    - Decide which group to use as set for test data.
    - Take the remaining groups as a training data set.
    - Fit a model on the training set and evaluate it on the test set.
    - Retain the evaluation score and discard the model.
4. Summarize the model using the sample of model evaluation score.

Results

In the following section, we will present the results of our regression models used on both the Franke function and terrain data, and discuss their performance based on the evaluation metrics.

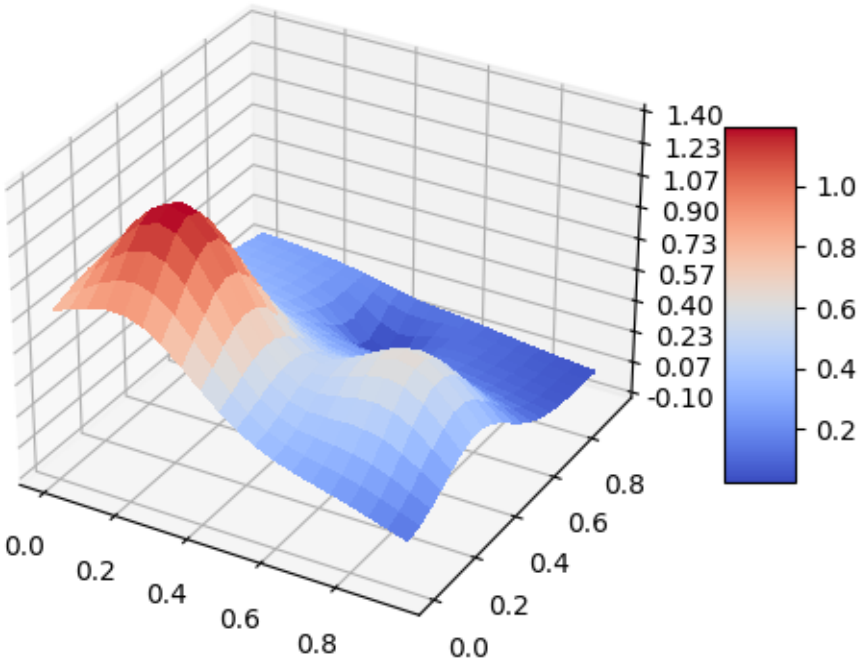
The code used in this section is provided in:
`https://github.com/camilldb/FYS-STK4155/tree/master/Project1`

The Franke function

We will first study how to fit polynomials to a specific two-dimensional function called Franke's function. This is a function which has been widely used when testing various interpolation and fitting algorithms. The Franke function, which is a weighted sum of four exponentials reads as follows

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

The function will be defined for $x, y \in [0, 1]$.



Ordinary Least Squares (OLS)

Our first step will be to perform an OLS regression analysis of this function, trying out a polynomial fit with an x and y dependence of the form $[x, y, x^2, y^2, xy, \dots]$. Since the data in one sense is scaled to a particular domain for the input values, I will not scale the data.

We will generate our own dataset for a function FrankeFunction(x, y) with $x, y \in [0, 1]$. The function $f(x, y)$ is the Franke function. I will explore also the addition of an added stochastic noise to this function using the normal distribution $N(0, 1)$.

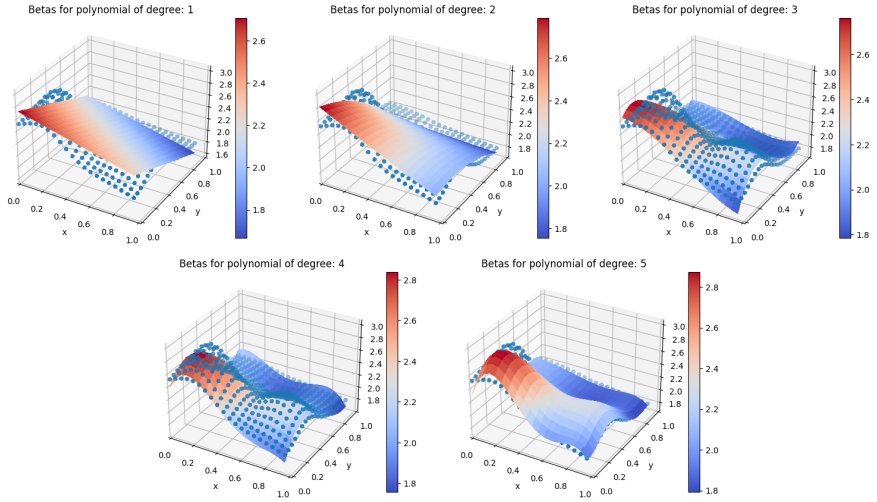
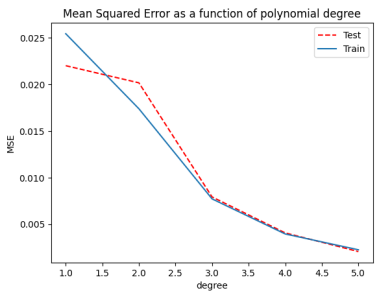
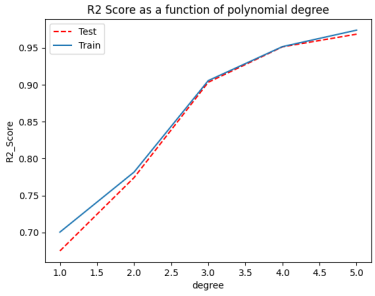


Figure 1: Betas with increasing order of polynomial degree plotted against the real data for Ordinary Least Squares, the real data is the dotted layer in the 3D-plot

I have also plotted the MSE and the R^2 Score as function of the polynomial degree which is shown in figure 2 below.
We can see that the MSE for the test set starts with a lower value than the training set. This make sense since the training set contains more data than the test set, and there will be more variations in this data set than the test data set. The MSE for both training and test set continuous to fall for increasing polynomial degree. After polynomial degree three the MSE fall by relatively small amounts as we fit successively higher-order-polynomial models. For increasing polynomial degree we would expect the MSE to increase again for some higher polynomial degree. The extra complexity of the higher-order polynomials allows the regression coefficients to be estimated very precisely. Almost too precisely. We are starting to pick up on patterns in the data that aren't really there, or at least patters that only are present in the training set, and not in the test set. This extra complexity makes our models great at predicting values it has



(a) Mean Squared Error



(b) R^2 Score

Figure 2: MSE and R^2 Score plotted as function of polynomial degree for Ordinary Least Squares

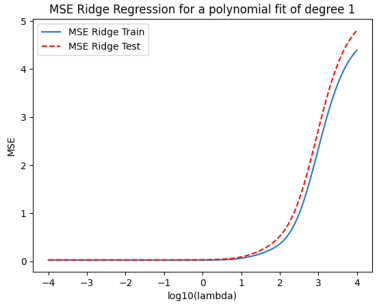
already seen before, but not so great at predicting new data. This is overfitting, and it's why we should use the test MSE to evaluate our models, rather than training MSE!

If we analyse the R^2 Score, we can see that the training set R^2 scores starts a bit higher than the R^2 Score. Then both values increase with higher order of polynomial degree. But after the third degree polynomial the R^2 Score increase by relatively small amounts as we fit successively higher-order-polynomial models. Which we would expect after analysing the MSE curve.

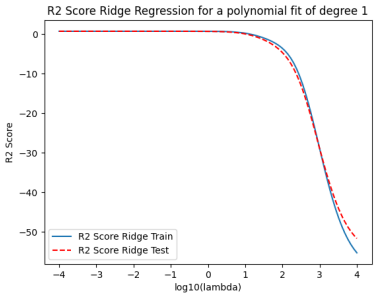
Since we here are only evaluating the MSE and R^2 score for polynomial degree up to fifth order. We can choose the fifth order polynomial degree fitting, as this has the lowest MSE and the highest R^2 Score without overfitting the data.

Ridge Regression

I have now performed the same analysis as I did in the previous section for both MSE and R^2 Score but now for different values of λ. In this project I have used hundred different λ values in the interval $[-4, 4]$.



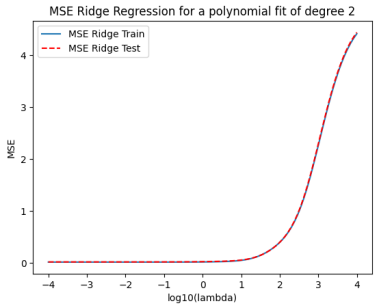
(a) Mean Squared Error



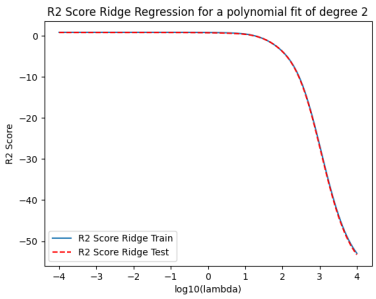
(b) R^2 Score

Figure 3: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 1

Looking at figure 3 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 0.02532 and 0.60202 repectively. Which means that if we want to use Ridge regression as our model we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we should choose Ordinary least squares as our regression model.



(a) Mean Squared Error

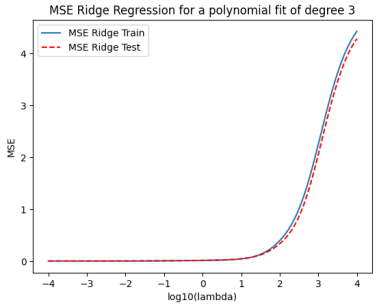


(b) R^2 Score

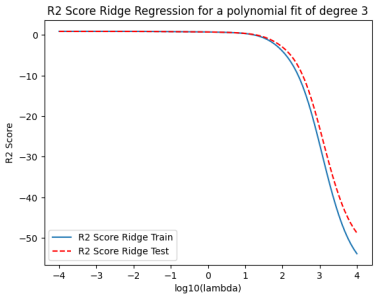
Figure 4: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 2

Looking at figure 4 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 0.01792 and 0.7774 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we

should choose Ordinary least squares as our regression model.



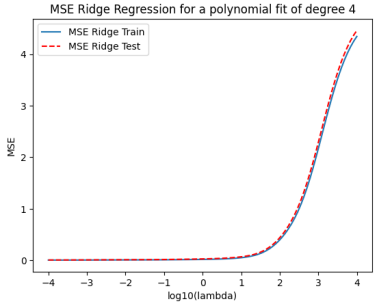
(a) Mean Squared Error



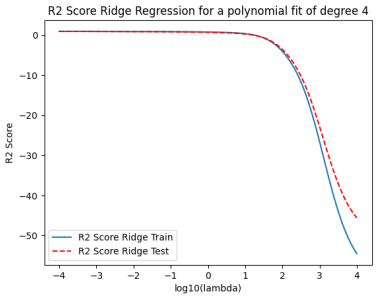
(b) R^2 Score

Figure 5: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 3

Looking at figure 5 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 0.00598 and 0.92947 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we should choose Ordinary least squares as our regression model.



(a) Mean Squared Error

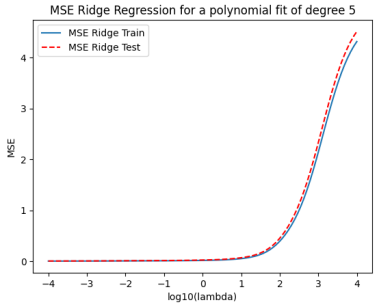


(b) R^2 Score

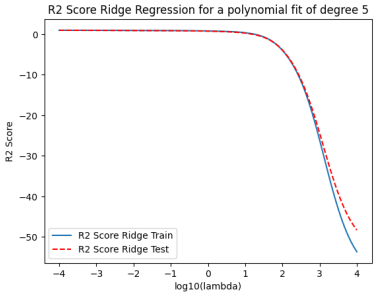
Figure 6: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 4

Looking at figure 6 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 0.004089 and 0.95133 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we should choose Ordinary least squares as our regression model.

Looking at figure 7 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 0.00295 and 0.95663 respectively. Which

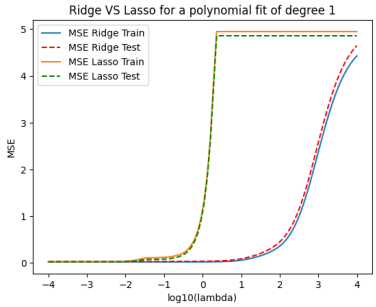


(a) Mean Squared Error

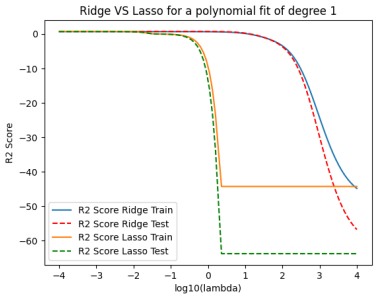


(b) R^2 Score

Figure 7: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 5



(a) Mean Squared Error



(b) R^2 Score

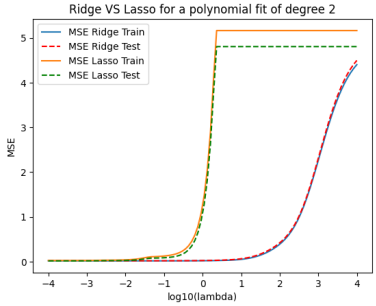
Figure 8: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 1

means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we should choose Ordinary least squares as our regression model. We can see that in general for all polynomial degrees from one to fifth order. We would choose Ordinary Least Squares as our model if $\lambda \in [1, 4]$, and choose Ridge regression as our model for $\lambda \in [-4, 1]$.

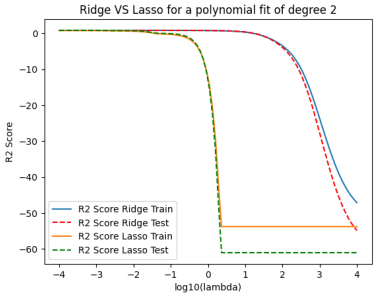
Lasso Regression

Now I have performed the same analysis as I did in the previous exercise for both MSE and R^2 Score with the same values of λ. In this project I have used hundred different λ values in the interval $[-4, 4]$. Using this results and comparing them to the results from Ridge regression for different polynomial degrees.

Looking at figure from 8 to 12 we can see that the Lasso regression depends

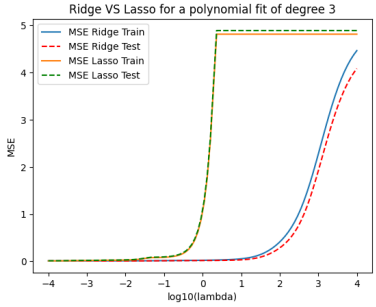


(a) Mean Squared Error

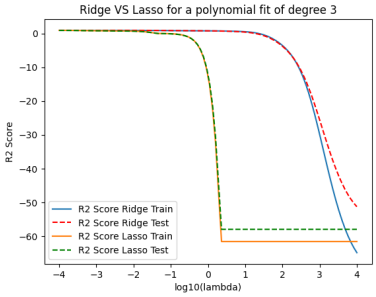


(b) R^2 Score

Figure 9: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 2

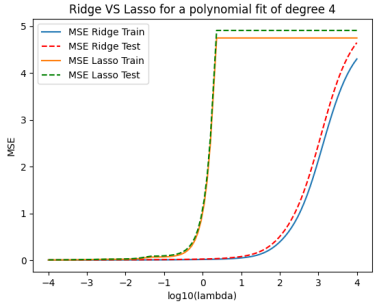


(a) Mean Squared Error

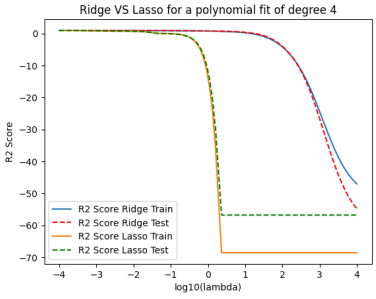


(b) R^2 Score

Figure 10: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 3

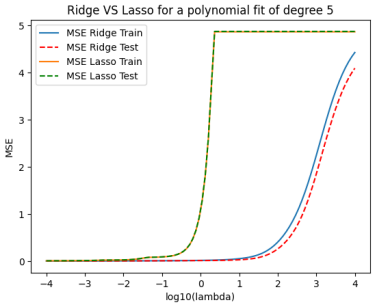


(a) Mean Squared Error

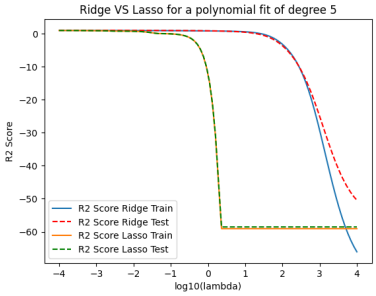


(b) R^2 Score

Figure 11: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 4



(a) Mean Squared Error



(b) R^2 Score

Figure 12: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 5

more on lambda than Ridge regression does. Meaning that higher values of λ results in higher MSE and lower R^2 score for every polynomial degree from one to five. The Lasso regression may give a better fit than Ridge regression for $\lambda \in [-4, -2]$ by looking at the figures above. While Ridge regression will be the preferred model for $\lambda \in [-2, 1]$. While OLS will be the best model with $\lambda > 1$.

Expectation value and variance

This section deals with various mean values and variances in linear regression method. If we assume that there exists a continuous function $f(\boldsymbol{x})$ and a normal distributed error $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ which describes our data

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

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

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

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

We can now show that the expectation value of $\boldsymbol{y}$ for a given element i. Since we know that

$$y_i = f(x_i) + \epsilon_i,$$

we can take the expectation of y_i

$$\begin{aligned} \mathbb{E}(y_i) &= \mathbb{E}[f(x_i) + \epsilon_i] \\ &= \mathbb{E}[f(x_i)] + \mathbb{E}(\epsilon_i) \end{aligned}$$

Since $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ we know that $\mathbb{E}(\epsilon_i) = 0$, hence

$$\mathbb{E}(y_i) = \mathbb{E}[f(x_i)]$$

Recall now that the function f is approximated by $\tilde{\boldsymbol{y}}$, where

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

Then we can write $f(x_i)$ as

$$\tilde{y}_i = \sum_j x_{ij}\beta_j$$

which is a continuous function which depends linearly on this unknown parameters $\boldsymbol{\beta}^T = [\beta_0, \beta_1, \beta_2, ..., \beta_j]$.Then we have

$$\begin{aligned}\mathbb{E}[f(x_i)] = \mathbb{E}(y_i) = \tilde{y}_i &= \sum_j x_{ij}\beta_j \\ &= x_{i,0}\beta_0 + x_{i,1}\beta_1 + x_{i,2}\beta_2 + ... + x_{,n-1}\beta_{n-1} \\ &= \mathbf{X}_{i,*}\boldsymbol{\beta}\end{aligned}$$

where $\mathbf{X}_{i,*}$ represents the i-th row of a matrix $\mathbf{X}$.

Now we are going to show that its variance is

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

We define the variance as

$$\begin{aligned}\boldsymbol{Var}(y_i) &= \mathbb{E}\{(y_i - \mathbb{E}(y_i))^2\} \\ &= \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2\end{aligned}$$

Now using that $y_i = \mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i$ and $\mathbb{E}(y_i) = \mathbf{X}_{i,*}\boldsymbol{\beta}$

$$\begin{aligned}\boldsymbol{Var}(y_i) &= \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \\ &= \mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i)^2] - (\mathbf{X}_{i,*}\boldsymbol{\beta})^2\end{aligned}$$

Using that

$$(\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i)^2 = (\mathbf{X}_{i,*}\boldsymbol{\beta})^2 + 2\epsilon_i\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i^2$$

we have

$$\mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i)^2] - (\mathbf{X}_{i,*}\boldsymbol{\beta})^2 = (\mathbf{X}_{i,*}\boldsymbol{\beta})^2 + 2\mathbb{E}(\epsilon_i)\mathbf{X}_{i,*}\boldsymbol{\beta} + \mathbb{E}(\epsilon_i^2) - (\mathbf{X}_{i,*}\boldsymbol{\beta})^2$$

Since $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$, meaning that $\mathbb{E}(\epsilon_i) = 0$, then

$$(\mathbf{X}_{i,*}\boldsymbol{\beta})^2 + 2\mathbb{E}(\epsilon_i)\mathbf{X}_{i,*}\boldsymbol{\beta} + \mathbb{E}(\epsilon_i^2) - (\mathbf{X}_{i,*}\boldsymbol{\beta})^2 = \mathbb{E}(\epsilon_i^2)$$

Still using the fact that $\mathbb{E}(\epsilon_i) = 0$, we can write

$$\boldsymbol{Var}(\epsilon_i) = \mathbb{E}\{[\epsilon_i - \mathbb{E}(\epsilon_i)^2]\} = \mathbb{E}(\epsilon_i^2) - [\mathbb{E}(\epsilon_i)^2] = \mathbb{E}(\epsilon_i^2)$$

Hence

$$\boldsymbol{Var}(y_i) = \mathbb{E}(\epsilon_i^2) = \boldsymbol{Var}(\epsilon) = \sigma^2.$$

Hence, $y_i \sim N(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$, that is $\boldsymbol{y}$ follows a normal distribution with mean value $\boldsymbol{X\beta}$ and variance σ^2.

With the OLS expressions for the optimal parameters $\hat{\boldsymbol{\beta}}$ we are going to show that

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}.$$

To find the optimal parameter $\hat{\beta}$ we first need to define the cost/loss function for OLS. The cost function is a way to measure the quality of our model, we can define this function as

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})\}$$

To find $\hat{\beta}$ we need to derivative of $C(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}^T$

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

Solving this equation with respect to $\boldsymbol{\beta}$ we get

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

Using this in the equation above we get

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

Using that $\mathbb{E}(\boldsymbol{y}) = \boldsymbol{X\beta}$ we get

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

and since is $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X} = \mathbb{I}$, we have that

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$$

which is what we wanted to show.
Finally we are going to show that the variance of $\boldsymbol{\beta}$ is

$$\boldsymbol{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2\,(\mathbf{X}^T\mathbf{X})^{-1}.$$

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

$$\boldsymbol{Var}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\{[\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}})]^2\}$$

To ensure that $\hat{\boldsymbol{\beta}}^2$ gives an answer we multiply $\hat{\boldsymbol{\beta}}$ with its own transposed $\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T$. Then we get an answer when $\hat{\boldsymbol{\beta}}$ is not a squared matrix. Using this in the equation above we can write

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

Now using that $\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ and that $\mathbb{E}(\boldsymbol{\beta}) = \boldsymbol{\beta}$

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

Where

$$\begin{aligned}
\mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^T\} &= \mathbb{E}[(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})^T] \\
&= (\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T) + 2\boldsymbol{X}\boldsymbol{\beta}\mathbb{E}(\boldsymbol{\epsilon}) + \boldsymbol{\epsilon}^2 \\
&= (\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T) + \boldsymbol{Var}(\boldsymbol{\epsilon}) \\
&= (\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T) + \sigma^2
\end{aligned}$$

Since $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$. Then we have

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

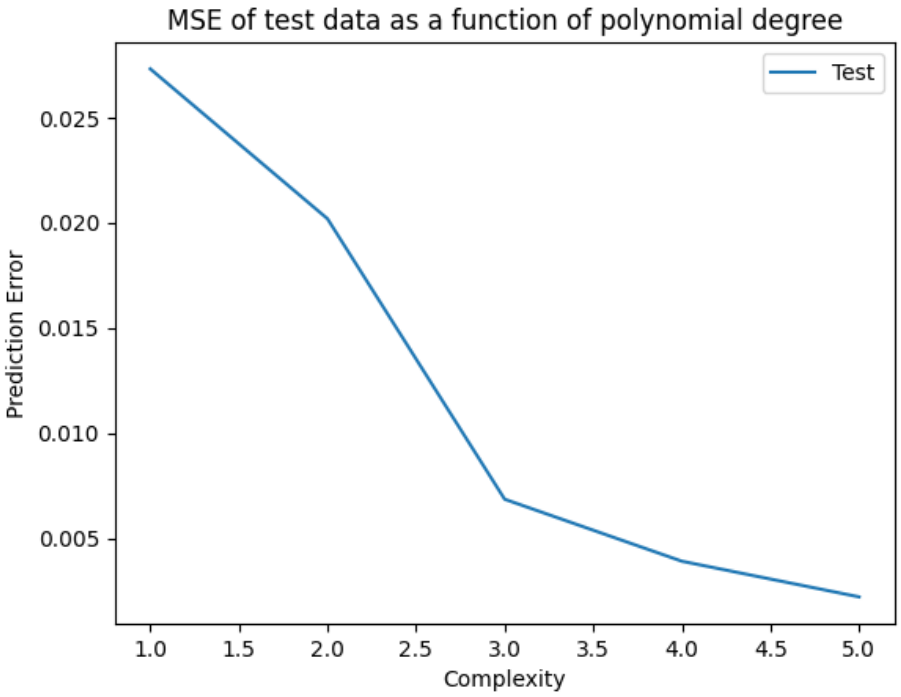
Which is what we wanted. We can use the last expression when we define a so-called confidence interval for the parameters β. A given parameter β_j is given by the diagonal matrix element of the above matrix.

Bias-variance tradeoff and resampling techniques

In this section our aim is to study the bias-variance trade-off by implementing the bootstrap resampling technique. We will only use the simpler ordinary least squares here.

With a code which does OLS and includes resampling techniques, we will now discuss the bias-variance trade-off in the context of continuous predictions such as regression. However, many of the intuitions and ideas discussed here also carry over to classification tasks and basically all Machine Learning algorithms.

Before I perform an analysis of the bias-variance trade-off on my test data, I will make a figure similar to Fig. 2.11 of Hastie, Tibshirani, and Friedman [2].



We can see that the MSE decreases with the order of complexity, but the higher complexity it decreases less and less for each degree. With this result we move on to the bias-variance trade-off analysis.

Consider a data set $\mathcal{L}$ consisting of the data $\mathbf{X}_{\mathcal{L}} = \{(y_j, \boldsymbol{x}_j), j = 0 \dots n-1\}$. The true data is generated from a noisy model

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

Here ϵ is normally distributed with mean zero and standard deviation σ^2.

In our derivation of the ordinary least squares method we defined then an approximation to the function f in terms of the parameters $\boldsymbol{\beta}$ and the design matrix $\boldsymbol{X}$ which embody our model, that is $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$.

The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the mean squared error via the so-called cost function

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

Here the expected value $\mathbb{E}$ is the sample value.

I am now going to show that you can rewrite this in terms of a term which contains the variance of the model itself (the so-called variance term), a term which measures the deviation from the true data and the mean value of the model (the bias term) and finally the variance of the noise. That is, I am going to show that

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

Using the information we have from above, we can write.

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

Solving the terms one by one, using that $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$:

$$\begin{aligned}\mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon})^2] &= \mathbb{E}[\boldsymbol{f}^2] + 2\mathbb{E}[\boldsymbol{f}\boldsymbol{\epsilon}] + \mathbb{E}[\boldsymbol{\epsilon}^2] \\ &= \boldsymbol{f}^2 + \sigma^2\end{aligned}$$

$$\begin{aligned}\mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon})\tilde{\boldsymbol{y}}] &= \mathbb{E}[\boldsymbol{f}\tilde{\boldsymbol{y}}] + \mathbb{E}[\boldsymbol{\epsilon}\tilde{\boldsymbol{y}}] \\ &= \boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}] + \mathbb{E}[\tilde{\boldsymbol{y}}]\mathbb{E}[\boldsymbol{\epsilon}] \\ &= \boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}]\end{aligned}$$

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

Then we can rewrite the above expression

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \boldsymbol{f}^2 + \sigma^2 - 2\boldsymbol{f}\mathbb{E}[\tilde{\boldsymbol{y}}] + \boldsymbol{Var}[\tilde{\boldsymbol{y}}] + (\mathbb{E}[\tilde{\boldsymbol{y}}])^2 \\ &= \mathbb{E}\left[(\boldsymbol{y}-\mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + \mathrm{Var}\left[\tilde{\boldsymbol{y}}\right] + \sigma^2 \\ &= (\mathrm{Bias}[\tilde{y}])^2 + \mathrm{var}[\tilde{y}] + \sigma^2\end{aligned}$$

with

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

and

$$\mathrm{var}[\tilde{f}] = \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2.$$

The three terms represent the square of the bias of the learning method, which can be thought of as the error caused by the simplifying assumptions built into the method. The second term represents the variance of the chosen model and finally the last terms is variance of the error $\boldsymbol{\epsilon}$.

I have also perform a bias-variance analysis of the Franke function by studying the MSE value as function of the complexity of your model and explored different number of bootstrap samples. Meaning the number of bootstrap samples generated during each iteration of the analysis. In a bias-variance tradeoff analysis, resampling is used to estimate how the model's performance varies with different training datasets. Each bootstrap sample is created by randomly selecting data points with replacement from the original dataset. By varying this parameter it determines how many times this resampling process is repeated, and therefore, how many different training datasets are created for each polynomial degree. By varying the parameter, I can observe how the analysis results change based on different resampling strategies. Typically, a larger

number of bootstrap samples will provide more stable estimates of the model's performance, but it can also be more computationally intensive.

I have explored with three different number of bootstrap samples (10, 50, and 100) while keeping the maximum polynomial degree fixed at 5. This allows me to compare how the bias and variance tradeoff varies for the same polynomial degree but with different resampling strategies, giving me insights into the model's stability and performance.

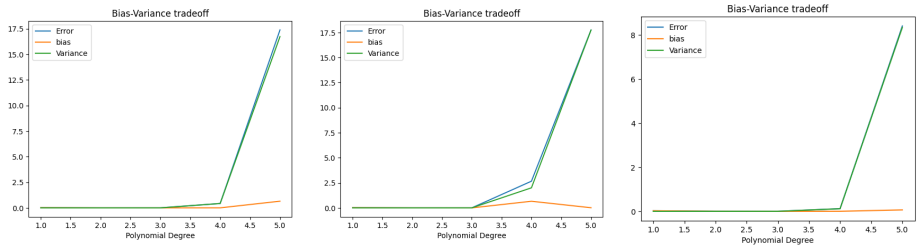


Figure 13: Bias-variance tradeoff for different number of bootstrap samples, to the left 10 bootstrap samples, middle 50 bootstrap samples, right 100 bootstrap samples are used

Lets look at the bias, variance, and the MSE for the right figure:

Degree	1	2	3	4	5
Bias	0.02578	0.00327	0.000281	0.003545	0.065383
Variance	0.0003761	0.00022	0.001054	0.118482	8.342
Error (MSE)	0.02615	0.003494	0.001336	0.1220	8.40766

From this we can see that when we increase the number of bootstrap samples the Error (MSE) and the Variance of the model becomes more stable. And if we analyse the figure to the right in figure 13, where we have used 100 bootstrap samples we can see that the model variance and error starts to increase after the third polynomial degree. Meaning in the case of Ordinary Least Squares we would benefit from using a polynomial fit of lower order than four.

Cross-validation K-folds

We can see from figure 14 that for OLS MSE is lowest polynomial degree one and three. The MSE for Ridge regression is stable for all values of λ, using the same lambdas chosen for Ridge and Lasso regression for the Franke function. Meaning hundred values of λ between $[-4, 4]$. For the Lasso regression we can see that the MSE is at it lowest at $\lambda = -2$ but increases for $\lambda > -2$.

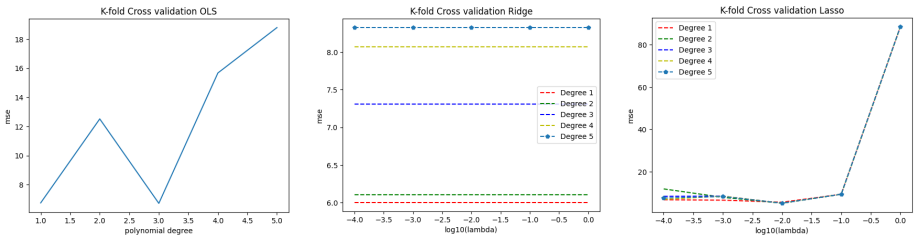
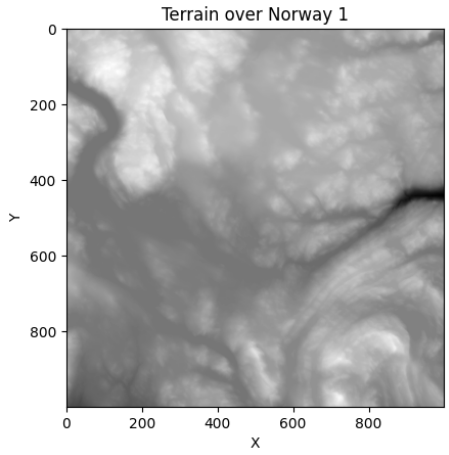


Figure 14: K-fold cross validation for OLS, Ridge and Lasso with $k = 5$

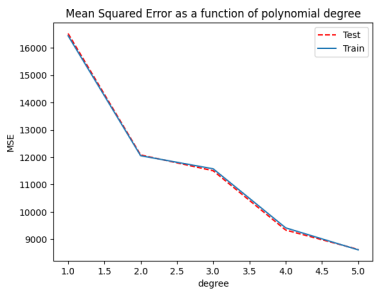
Terrain data

We have now used different regression methods and resampling techniques on the Franke function defined above. It is now time to test these functions on real data. We will essentially repeat all we did above in this section only terrain data. More specifically the terrain from a region close to Stavanger in Norway. Below you can see both the 2D-plot of the terrain data, and a 3D-plot. I have chosen to use only $(1000, 1000)$ point of the original data set consisting of $(3601, 1801)$ point.

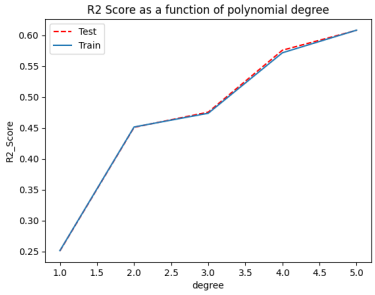


Ordinary Least Squares

We can see from this plots in figure 15 that both the training data and the test data benefit from a higher polynomial degree. We would expect this to drop further to a certain polynomial degree and than increase again. But this is something we can't see when we just plot for polynomial degree up fifth order. Looking at the values of the MSE, we would probably benefit from scaling this data.



(a) Mean Squared Error

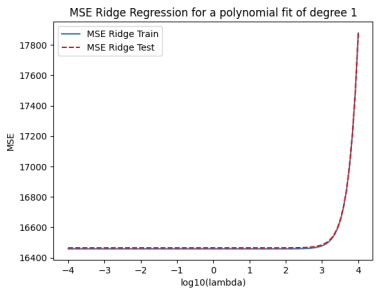


(b) R^2 Score

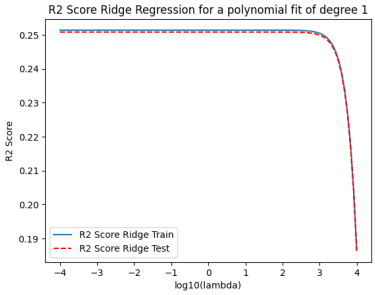
Figure 15: MSE and R^2 Score plotted as function of polynomial degree for the Terrain data

Ridge Regression

I have now performed the same analysis as I did in the previous section for the Franke function for both MSE and R^2 Score but now for different values of λ. In this project I have used hundred different λ values in the interval $[-4, 4]$ for both the Franke function and the terrain data.



(a) Mean Squared Error

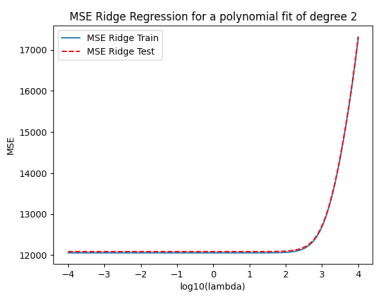


(b) R^2 Score

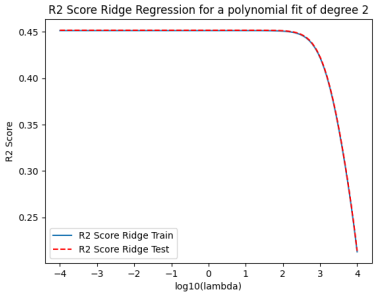
Figure 16: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 1

Looking at figure 16 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 16524 and 0.2508 respectively. Which means that if we want to use Ridge regression as our model we should choose $\lambda \in [-4, 2]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [2, 4]$ we should choose Ordinary least squares as our regression model.

Looking at figure 17 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 12082 and 0.45093 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 2]$



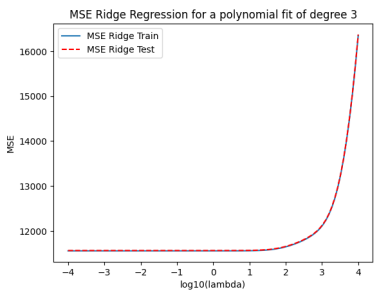
(a) Mean Squared Error



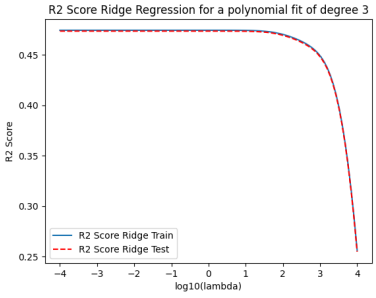
(b) R^2 Score

Figure 17: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 2

to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [2, 4]$ we should choose Ordinary least squares as our regression model.



(a) Mean Squared Error



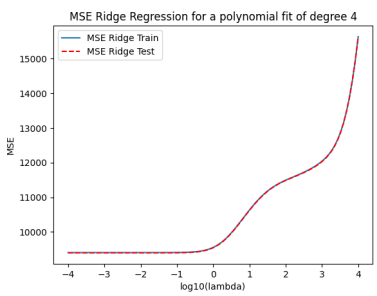
(b) R^2 Score

Figure 18: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 3

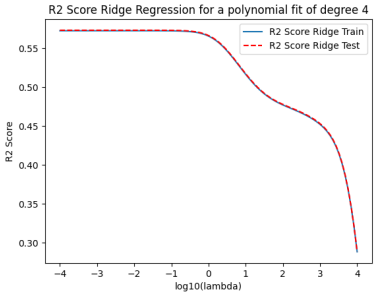
Looking at figure 18 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 11504 and 0.92947 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [1, 4]$ we should choose Ordinary least squares as our regression model.

Looking at figure 19 and comparing this results to the MSE and R^2 Score for Ordinary least Squares, which is 9334 and 0.5757 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, 0]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [0, 4]$ we should choose Ordinary least squares as our regression model.

Looking at figure 20 and comparing this results to the MSE and R^2 Score

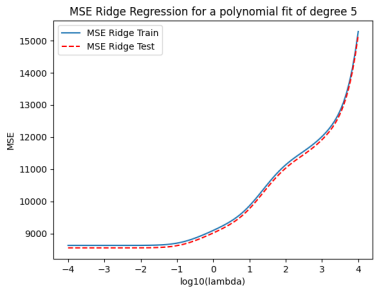


(a) Mean Squared Error

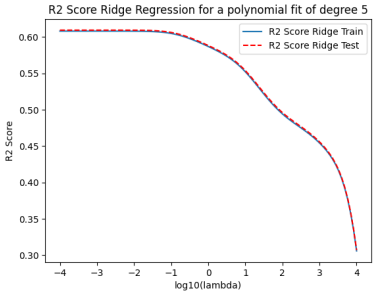


(b) R^2 Score

Figure 19: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 4



(a) Mean Squared Error



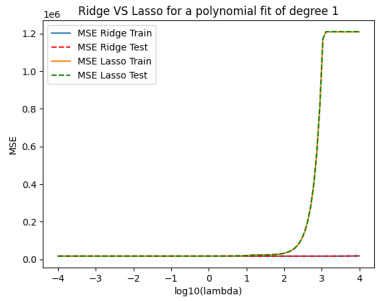
(b) R^2 Score

Figure 20: MSE and R^2 Score plotted as function of lambda for Ridge regression with polynomial degree 5

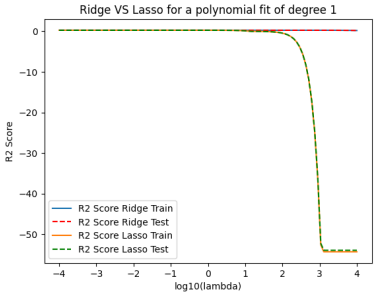
for Ordinary least Squares, which is 8625 and 0.6080 respectively. Which means that if we want to use Ridge regression in this case we should choose $\lambda \in [-4, -1]$ to get a equal or better MSE and R^2 Score then for OLS. When $\lambda \in [-1, 4]$ we should choose Ordinary least squares as our regression model. We can see that in general for all polynomial degrees from one to fifth order. We would choose Ordinary Least Squares as our model if $\lambda \in [-1, 4]$, and choose Ridge regression as our model for $\lambda \in [-4, -1]$.

Lasso Regression

Looking at figure from 21 to 25 we can see that the Lasso regression depends more on lambda than Ridge regression does. Meaning that higher values of λ results in higher MSE and lower R^2 score for every polynomial degree from one to five. The Lasso regression may give a better fit than Ridge regression for $\lambda \in [-4, 1]$ by looking at the figures above. While Ridge regression will be

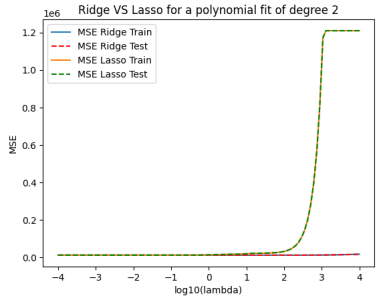


(a) Mean Squared Error

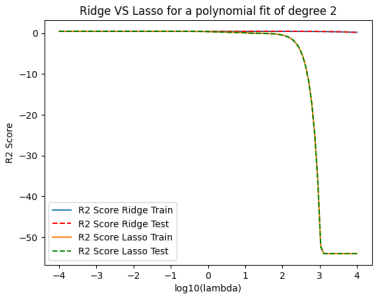


(b) R^2 Score

Figure 21: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 1



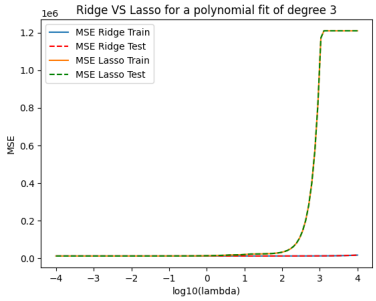
(a) Mean Squared Error



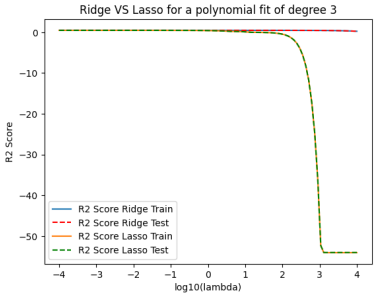
(b) R^2 Score

Figure 22: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 2

the preferred model between Lasso and Ridge for $\lambda \in [1, 4]$. But for all the Regression models OLS will be the best model with $\lambda > 1$.

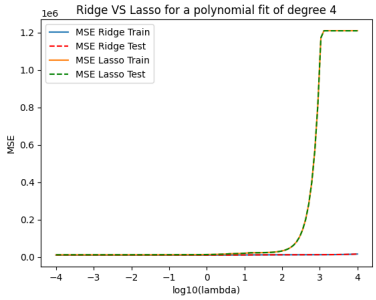


(a) Mean Squared Error

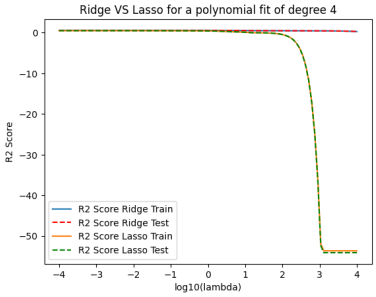


(b) R^2 Score

Figure 23: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 3

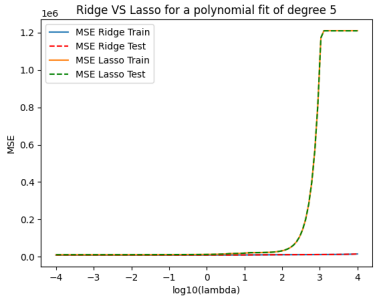


(a) Mean Squared Error

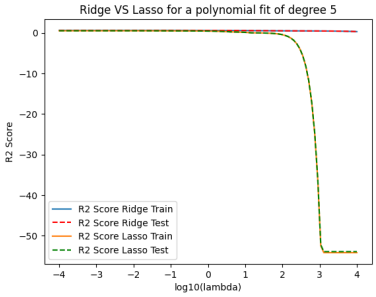


(b) R^2 Score

Figure 24: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 4



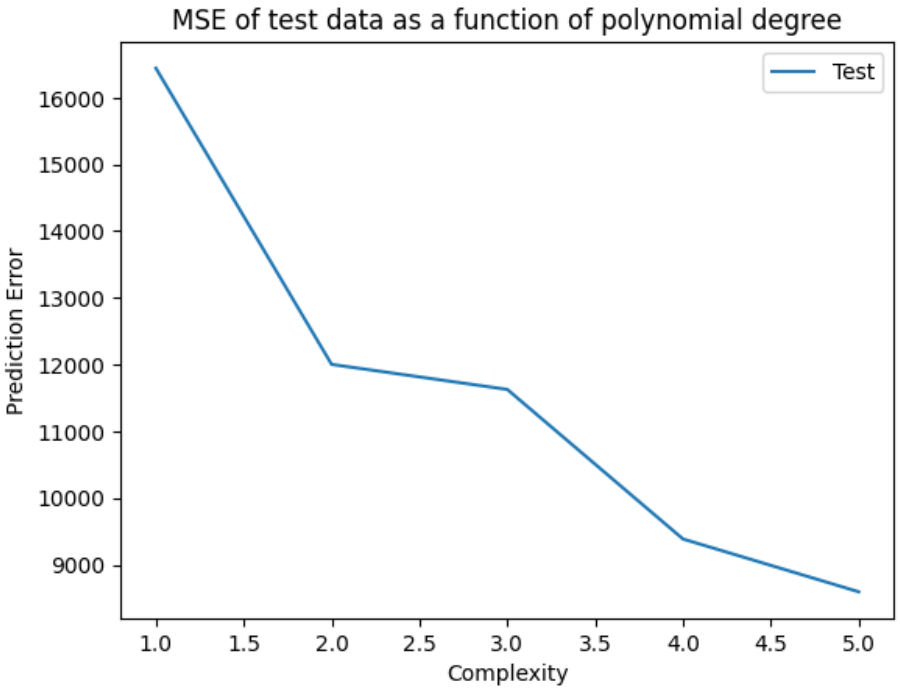
(a) Mean Squared Error



(b) R^2 Score

Figure 25: MSE and R^2 Score plotted as function of lambda for Ridge and Lasso regression with polynomial degree 5

Bias-variance tradeoff and resampling techniques



We can see that the MSE decreases with the order of complexity, but the higher complexity it decreases less and less for each degree. With this result we move on to the bias-variance trade-off analysis

Figure 26: Bias-variance tradeoff for different number of bootstrap samples, to the left 10 bootstrap samples, middle 50 bootstrap samples, right 100 bootstrap samples are used

Lets look at the bias, variance, and the MSE for the right figure:

Degree	1	2	3	4	5
Bias	16464.7	12088.46	11514.62	9386.05	8610.77
Variance	0.06365	0.099515	0.15551	0.19708	0.261738
Error (MSE)	16464.76	12088.56	11514.78	9386.25	8611.036

From this we can see that when we increase the number of bootstrap samples the Error (MSE) of the model becomes more stable, in this case the variance is almost zero for every polynomial degree. And if we analyse the figures in figure 26, we can see that we benefit in from a higher order polynomial fit for this data set.

Cross-validation K-folds

To run the code made for K-folds cross validation for $k = 5$ I needed to use only a part of the Terrain data as used in the previous parts. In this section a (100,100) part of the data.

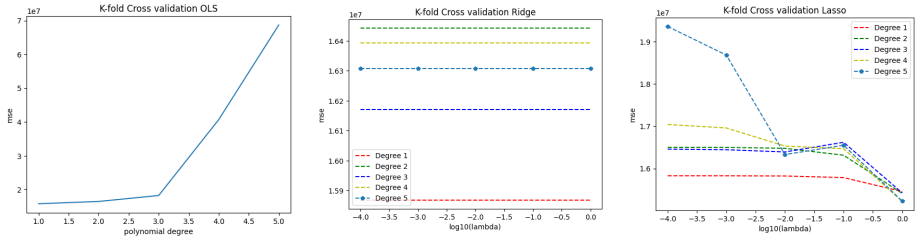


Figure 27: K-fold cross validation for OLS, Ridge and Lasso with $k = 5$

We can see from figure 27 that for OLS MSE is lowest at polynomial degree one and from here on the MSE increases with the polynomial order. The MSE for Ridge regression is stable for all values of λ, using the same lambdas chosen for Ridge and Lasso regression for the Franke function. Meaning hundred values of λ between $[-4, 4]$. For the Lasso regression we can see that the MSE is at it lowest for $\lambda = 0$ but increases for $\lambda < 0$.

Conclusions

When we used our generated data from Franke's function the Ordinary Least Squares regression provided us with the best R^2 score and lowest MSE when we used a polynomial degree of 5. We could probably expect a increase in the MSE with higher order polynomials. When comparing the Ridge regression with OLS, we would choose the OLS for $\lambda > 1$. With $\lambda < 1$ Ridge regression would be preferable. When adding the Lasso regression we could see that this regression method would be a good choice if $\lambda < -2$.

For the real terrain data, we could see that the Ordinary Least Squares regression provided us with the best R^2 score and lowest MSE when we used a polynomial degree of 5. We could probably expect a increase in the MSE with higher order polynomials. When comparing the Ridge regression with OLS, we would choose the OLS for $\lambda > -1$. With $\lambda < -1$ Ridge regression would be preferable. When adding the Lasso regression we could see that this regression method would be a good choice if $\lambda < 1$.

Bibliography

[1] Morten Hjorth-Jensen. Lecture Notes in FYS-STK4155. Applied data analysis and Machine Learning. 2023.

[2] Trevor Hastie, Robert Tibshirani, and JH Friedman. The elements of statistical learning: data mining, inferece, and prediction. 2009.