# Project 1 - Regression analysis and resampling methods

Arangan Subramaniam & Krithika Gunesegaran

October 2024

# Contents

**1 Introduction** **3**

**2 Method** **3**

2.1 Linear Regression . . . . . . . . . . 4

2.2 Ordinary Least Square . . . . . . . . . . 5

2.3 Ridge Regression . . . . . . . . . . 7

2.4 Lasso Regression . . . . . . . . . . 7

2.5 Bias-Variance Tradeoff . . . . . . . . . . 8

2.6 Resampling . . . . . . . . . . 9

2.6.1 Bootstraping . . . . . . . . . . 9

2.6.2 K-fold Cross-Validation . . . . . . . . . . 10

**3 Implementation** **10**

3.1 Franke Function . . . . . . . . . . 10

3.2 Terrain Data . . . . . . . . . . 11

**4 Results** **12**

4.1 Franke Function . . . . . . . . . . 12

4.2 Terrain Data . . . . . . . . . . 18

**5 Discussion** **19**

5.1 Comparison OLS, Ridge and Lasso . . . . . . . . . . 19

5.2 Bias Variance and Resampling . . . . . . . . . . 20

5.3 Terrain Data . . . . . . . . . . 20

**6 Conclusion** **21**

**7 Appendix** **22**

**8 Acknowledgement** **22**

**9 References** **22**

**Abstract**

This project investigates the use of regression techniques to predict both synthetic (Franke Function) and real-world terrain data, addressing challenges such as overfitting and model complexity. Ordinary Least Squares (OLS), Ridge, and Lasso regression are applied to the Franke Function and real terrain data to optimize model performance, with a focus on the bias-variance tradeoff. We leverage resampling techniques such as bootstrapping and k-fold cross-validation to improve model evaluation. While OLS performed well on training data, Ridge regression showed superior generalization when evaluated with with cross-validation and bootstraping, making it the most reliable for both synthetic and real-world data. For the Franke function, OLS with a polynomial degree of polynomial degree of 5 provided the best fit, achieving a Mean Squared Error (MSE) of 0.0025. For terrain data, Ridge regression with a polynomial degree yielded the best results, with an MSE of 1.6.

## 1 Introduction

In this report, we explore different regression techniques, focusing on how they handle model complexity and overfitting, which are common challenges in data analysis. Regression methods like Ordinary Least Squares (OLS), Ridge, and Lasso are important tools for modeling relationships between variables, but they differ significantly in how they manage complexity, especially in complex datasets like terrain data. This analysis will help highlight the strengths and weaknesses of these models.

In this project, we apply these regression methods to two datasets: the Franke function and real world Digital Terrain data. We also investigate the effects of model complexity and regularization through bias variance trade-off and resampling techniques such as bootstrap and k-fold cross validation.

The report is organized as follows: We introduce the theoretical methods behind OLS, Ridge, Lasso regression and Resampling. Then, we present the implementation and application of these methods to the datasets. Following that, we discuss the results and compare model performance. Finally, we conclude by summarizing the findings and highlighting the most suitable model for complex data.

## 2 Method

This section provides an overview of the theoretical basis and the methodology used on this project. We will present various regression and resampling techniques, along with the mathematical expressions for the models used.

Regression modeling explains how the distribution of a random variable changes based on one or more other variables. The aim of regression analysis is to explore the relationship between the dependent variable $y_i$ and the independent variables $x_i$. This helps us understand the causal relationships and identify functional patterns, and make predictions or fits.

## 2.1 Linear Regression

Linear Regression analysis is a statistical method used to predict the value of a variable based on the value of another variable [7]. In general, the models are structured around a dataset $\{x_i, y_i\}$, where $i \in \{0, 1, 2, \dots, n-1\}$. Here, $x_i$ denotes the independent variables, while $y_i$ represents the dependent variables. This data can be described as,

$$y_i = f(x_i) + \epsilon_i \tag{1}$$

or in general

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{2}$$

Here, $\boldsymbol{y}$ is the observed dependent variable, and $f(\boldsymbol{x})$ is the model that approximates the relationship between $\boldsymbol{y}$ and the independent variables $\boldsymbol{x}$. $\boldsymbol{\epsilon}$ represents some noise which is normally assumed to be distributed via a normal probability distribution with zero mean value and a variance $\sigma^2$ [6].

When we approximate the function $f(\boldsymbol{x})$ using a polynomial of degree $n-1$, we express it as a linear combination of powers of $x_i$ with coefficients $\beta_j$. This gives us

$$y(x_i) = \sum_{j=0}^{n-1} \beta_j x_i^j + \epsilon_i \tag{3}$$

where $\sum_{j=0}^{n-1} \beta_j x_i^j$ the polynomial prediction for $y(x_i)$ and $\epsilon_i$ is the error in our approximation [3].

With $\beta \in \mathbb{R}^{p\times 1}$ and design matrix $\boldsymbol{X} = x_i^j$. We can rewrite Equation 3 as

$$\boldsymbol{y} = \boldsymbol{X}\beta + \boldsymbol{\epsilon} \tag{4}$$

$$\tilde{y} = \boldsymbol{X}\beta \tag{5}$$

To find the best parameters $\beta$ for our model, we need a way to measure how well the model's predictions $\tilde{y}_i$ match the actual data points $y_i$. The goal is to minimize the difference between these two sets of values.

To do this, we create a cost function $C(\beta)$ , which measures how far apart the actual values $y_i$ are from the predicted values $\tilde{y}_i$. This difference, which is the spread, tells us how inaccurate our predictions are [3].

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\left\{(\boldsymbol{y} - \tilde{\boldsymbol{y}})^T(\boldsymbol{y} - \tilde{\boldsymbol{y}})\right\} \tag{6}$$

We calculate the Mean Square Error as in Equation 8 to see how well the model performs. It takes the difference between each actual value $y_i$ and predicted value $\tilde{y}_i$, squares it and sums these squared differences across all data points. The result represents how well or poorly the model is performing. In order to minimize the error of the cost function $C(\beta)$, we have to solve the problem

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{n}\left\{(y-X\beta)^T(y-X\beta)\right\} \tag{7}$$

This could also be solved as MSE.

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

We will use the metrics MSE and $R^2$-score to evaluate the performance of each model. Ideally, we want MSE to be as small as possible, with lower values indicating a more accurate model.

The $R^2$-score, as defined in Equation 9, measures how well the model explains the variation in the data. An $R^2$ value of 1 indicates a perfect fit [3]. Therefore, a good model will have an MSE close to 0 and an $R^2$-score close to 1. If $R^2 - score = 0$, the model does not explain any variance and performs no better than predicting the mean. If $R^2 - score < 0$ indicates the model performs worse than predicting the mean, suggesting very poor predictive accuracy [3].

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

## 2.2 Ordinary Least Square

Ordinary Least Squares (OLS) is a method used in linear regression to find the best-fitting data by estimating the unknown parameters of the model [2]. It works by minimizing the differences between the observed data points and the values predicted by the linear model. By derivating the cost function from Equation 6 with respect to $\beta$, we arrive at the following expression:

$$\frac{\partial C(\beta)}{\partial\beta} = 0 = X^T(y-X\beta) \tag{10}$$

$$X^Ty = X^TX\beta \tag{11}$$

From this, we can define $\hat{\beta}_{\text{OLS}}$ in Equation 12, which provides the solution for the optimal parameter $\hat{\beta}_{\text{OLS}}$.

$$\hat{\beta}_{\text{OLS}} = (X^TX)^-1X^Ty, \tag{12}$$

By minimizing the cost function, we can find the optimal parameter $\hat{\beta}_{\text{OLS}}$ that gives the best-fitting model for the given data. This means we can rewrite our model from Equation 5 as

$$\tilde{y} = X\hat{\beta}_{\text{OLS}} = X(X^TX)^{-1}X^Ty \tag{13}$$

The expected value of $y$ from Equation 1, for a given element $i$, where $X\beta$ is non-stochastic, is expressed in Equation 14 for the $i$-th element of $y$.

$$\begin{aligned} y &= f(x) + \epsilon = X\beta + \epsilon \\ y_i &= \sum_{j=0}^{p-1} x_{ij}\beta_j + \epsilon_i = X_{i,*}\beta + \epsilon_i \\ \mathbb{E}[y_i] &= \mathbb{E}[X_{i,*}\beta] + \underbrace{\mathbb{E}[\epsilon_i]}_{=0} = X_{i,*}\beta \end{aligned} \tag{14}$$

The variance of $y$ for a given element $i$ is expressed as:

$$\begin{aligned} \mathrm{Var}(y_i) &= \mathbb{E}\left[(y_i - \mathbb{E}(y_i))^2\right] \\ &= \mathbb{E}\left[(X_{i,*}\beta + \epsilon_i)^2\right] - (X_{i,*}\beta)^2 \\ &= (X_{i,*}\beta)^2 + \underbrace{2\mathbb{E}(\epsilon_i)X_{i,*}\beta}_{=0} + \mathbb{E}\left[\epsilon_i^2\right] - (X_{i,*}\beta)^2 \\ &= \mathbb{E}(\epsilon_i^2) = \sigma^2 \end{aligned} \tag{15}$$

Given that $y_i$ is normally distributed, we can determine the expectation of the optimal parameter $\hat{\beta}_{\mathrm{OLS}}$ from Equation 12.

$$\begin{aligned} \mathbb{E}[\hat{\beta}_{\mathrm{OLS}}] &= \mathbb{E}[(X^TX)^{-1}X^Ty] \\ &= (X^TX)^{-1}X^T\mathbb{E}[y] \\ &= (X^TX)^{-1}X^T\mathbb{E}[X\beta + \epsilon] \\ &= (X^TX)^{-1}X^TX\mathbb{E}[\beta] + \underbrace{(X^TX)^{-1}X^T\mathbb{E}[\epsilon]}_{=0} = \mathbb{E}[\beta] \end{aligned} \tag{16}$$

$$\mathbb{E}[\hat{\beta}_{\mathrm{OLS}}] = \beta \tag{17}$$

The variance of the optimal parameter $\hat{\beta}_{\mathrm{OLS}}$ is affected by both the design matrix and $\sigma^2$ as shown in Equation 19.

$$\begin{aligned} \mathrm{Var}(\hat{\beta}_{\mathrm{OLS}}) &= \mathbb{E}\left\{(\beta - \mathbb{E}(\beta))(\beta - \mathbb{E}(\beta))^T\right\} \\ &= \mathbb{E}\left\{(X^TX)^{-1}X^Ty - \beta\right\} \times \left\{(X^TX)^{-1}X^Ty - \beta\right\}^T \\ &= (X^TX)^{-1}X^T\mathbb{E}\left\{yy^T\right\}X(X^TX)^{-1} - \beta\beta^T \\ &= (X^TX)^{-1}X^T\mathbb{E}\left\{(X\beta + \epsilon)(X\beta + \epsilon)^T\right\}X(X^TX)^{-1} - \beta\beta^T \\ &= (X^TX)^{-1}X^T[X\beta\beta^TX^T + \underbrace{X\beta\mathbb{E}(\epsilon^T)}_{=0} + \underbrace{\mathbb{E}(\epsilon)\beta^TX^T}_{=0} + \underbrace{\mathbb{E}[\epsilon\epsilon^T]}_{=\sigma^2} - \beta\beta^T \\ &= (X^TX)^{-1}X^TX\beta\beta^TX^TX((X^TX)^{-1})^T + ((X^TX)^{-1}X^T\sigma^2X(X^TX)^{-1})^T - \beta\beta^T \\ &= \beta\beta^T + \sigma^2(X^TX)^{-1} - \beta\beta^T \end{aligned} \tag{18}$$

$$\mathrm{Var}(\hat{\beta}_{\mathrm{OLS}}) = \sigma^2(X^TX)^{-1} \tag{19}$$

## 2.3 Ridge Regression

Ridge Regression addresses the limitations of Ordinary Least Squares by adding a regularization term $\lambda$ to the cost function $C(\beta)$ in Equation 6 [4]. The parameter $\lambda$, controls the strength of the penalty. A larger $\lambda$ increases the penalty, pushing the coefficients closer to zero. This helps improve the model by reducing overfitting while aiming to keep the MSE as low as possible. The Ridge solution modifies the OLS solution by adding regularization. The cost function for Ridge Regression is given as

$$C(\beta) = \frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \tag{20}$$

The first term minimizes prediction error, helping the model fit the data more accurately. The second term penalizes the sum of the squared coefficients, reducing their size without setting any to exactly zero. This makes Ridge useful when you want to keep all features but reduce the effect of less important ones. This can also be written as

$$C_{\text{Ridge}}(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - X_i\beta)^2 + \lambda\sum_{j=1}^{n-1}\beta_j^2 \tag{21}$$

Similar to OLS, we can find the optimal parameter $\hat{\beta}_{\text{Ridge}}$ by minimizing the cost function.

$$\begin{aligned}
\frac{\partial C_{\text{Ridge}}(\beta)}{\partial\beta} &= 0 = -\frac{2}{n}(y - X\beta)^T X + 2\lambda\beta^T \\
&= \frac{2}{n}(\beta^T X^T X - y^T X) + 2\lambda\beta^T \\
0 &= \beta^T(X^T X + \lambda I) - y^T X \\
\beta^T &= y^T X(X^T X + \lambda I)^{-1} \\
\beta &= (X^T X + \lambda I)^{-1} X^T y
\end{aligned} \tag{22}$$

$$\hat{\beta}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T y \tag{23}$$

The term $X^T X$ shows how the data features are related to each other, and adding $\lambda I$ controls the models complexity by reducing large coefficients. This adjustment helps Ridge handle multicollinearity and prevent overfitting. $\hat{\beta}_{\text{Ridge}}$, shows how it uses the regularization term $\lambda$, to improve the models ability to generalize to new data.

## 2.4 Lasso Regression

Similar to Ridge Regression, the Least Absolute Shrinkage and Selection Operator (LASSO) addresses the limitations of OLS by adding a regularization term $\lambda$ to the cost function $C(\beta)$ from Equation 6 [4]. LASSO penalizes the absolute values of the regression coefficients, shrinking them and improving variable selection. When the penalty $\lambda$ is large enough, LASSO can suppresses some coefficients to be exactly zero [4]. The cost function for LASSO is given by

$$C(\mathbf{X}, \beta) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1 \tag{24}$$

Just like Ridge, the first term focuses on minimizing the prediction error, ensuring that the model fits the data well. But the second term adds the regularization term $\lambda$ , shrinking the coefficients and even force some to be exactly zero when $\lambda$ is large enough.

This means Lasso not only reduces the models complexity, like Ridge 20, but also automatically selects important features, keeping only the most relevant ones. This is especially useful when working with high-dimensional data. The difference between Ridge and Lasso is that in Ridge, the penalty $\lambda$ only shrinks, while in Lasso, the sum of the absolute values of the coefficients, can shrink to zero.

## 2.5 Bias-Variance Tradeoff

The bias-variance tradeoff is about finding a balance between two errors: Bias, which is the error when a model is too simple and underfits, and variance, which is the error when a model is too complex and overfits. The goal is to find a model that generalizes well to new data by minimizing both bias and variance.

We can mathematically express the bias-variance tradeoff by solving the expectation of the cost function from Equation 6.

$$\mathbb{E}\left[(y - \tilde{y})^2\right] = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{y}])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{y}])^2 + \sigma^2 \tag{25}$$

$$\begin{aligned}
\mathbb{E}[(y-\tilde{y})^2] &= \mathbb{E}[y^2] - 2\mathbb{E}[y\tilde{y}] + \mathbb{E}[\tilde{y}^2] \\
&= \mathbb{E}[y^2] - 2\mathbb{E}[y\tilde{y}] + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= \mathbb{E}[(f+\epsilon)^2] - 2\mathbb{E}[y\tilde{y}] + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= \mathbb{E}[f^2] + \underbrace{2\mathbb{E}[f\epsilon]}_{=0} + \underbrace{\mathbb{E}[\epsilon^2]}_{=\sigma^2} + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= \mathbb{E}[f^2] + \sigma^2 - 2\mathbb{E}[y\tilde{y}] + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= \mathbb{E}[f^2] + \sigma^2 - 2\mathbb{E}[(f+\epsilon)\tilde{y}] \\
&= \mathbb{E}[f^2] + \sigma^2 - 2\mathbb{E}[f\tilde{y}] + \underbrace{2\mathbb{E}[\epsilon\tilde{y}]}_{=0} + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= \mathbb{E}[f^2] + \sigma^2 - 2\mathbb{E}[f\tilde{y}] + (\mathrm{Var}[\tilde{y}] + (\mathbb{E}[\tilde{y}])^2) \\
&= f^2 - 2f\mathbb{E}[\tilde{y}] + (\mathbb{E}[y])^2 + \mathrm{Var}[\tilde{y}] + \sigma^2 \\
&= \mathbb{E}[f - \mathbb{E}[\tilde{y}]]^2 + \mathrm{Var}[\tilde{y}] + \sigma^2 \\
&= \underbrace{\mathbb{E}[y - \mathbb{E}[\tilde{y}]]^2}_{=Bias[\tilde{y}]} \mathrm{Var}[\tilde{y}] + \sigma^2 \\
&= \mathrm{Bias}[\tilde{y}] + \mathrm{Var}[\tilde{y}] + \sigma^2
\end{aligned} \tag{26}$$

Since the function $f$ is a non-stochastic variable, the same applies to $\tilde{y}$. The expectation can be understood as an average, allowing us to express $\mathrm{Var}[\tilde{y}]$ as $\frac{1}{n}\sum_i(\tilde{y}_i)$. Therefore, Equation 26 leads to the same result as Equation 25.

Bias$[\tilde{y}]$ calculates the difference between the predicted values $y$ and the true value $\tilde{y}$. A model with a high bias often underfits the data, meaning its too simple to capture the underlying patterns [5]. When a model has high bias, it leads to errors both in the train and test data.

Var$[\tilde{y}]$ calculates how much the models predictions would change if you trained it on different parts of the dataset. A model with a high variance will produce different predictions depending on the specific data it was trained on, while a model with low variance will produce more consistent predictions across different datasets. When the model has high variance it is too sensitive and complex and adds noise and other fluctuations to the train data and as a result, the model can overfit.

The noise $\epsilon$ is the random noise in the data that even a perfect model can not remove because it comes from factors outside the scope of the model. Even the most accurate model can only reduce bias and variance errors, but irreducible error $\epsilon$ will always be present because it is part of the data itself [5].

With high bias and low variance, the model underfits by oversimplifying the data, missing important patterns and making inaccurate predictions. With low bias and high variance, the model overfits the training data and performs poorly when applied to new data. With the bias-variance trade-off, we want to find the right balance between bias and variance to build models to generalize well to the unseen data.

.

## 2.6 Resampling

Resampling methods are techniques used to generate new samples from existing data to improve model performance. They are particularly useful with small datasets, as they help evaluate model accuracy and prevent overfitting. In this project, we will apply the two most common resampling methods, bootstrapping and cross-validation.

### 2.6.1 Bootstraping

Bootstrapping is a nonparametric method that uses random sampling with replacement to estimate properties of unknown data. It generates new datasets by resampling the original data, allowing observations to be selected more than once. These new datasets are used to calculate statistics like mean and variance, and the process is repeated many times to build a bootstrap distribution, which helps estimate confidence intervals and standard errors [8].

Bootstrapping leverages the Central Limit Theorem (CLT), which states that the average of a large number of independent, identically distributed variables will be approximately normally distributed. This makes bootstrapping useful for small datasets or when the data distribution is unknown, providing reliable estimates even in complex cases where traditional methods are ineffective.

#### 2.6.2 K-fold Cross-Validation

Cross-validation evaluates a models performance by testing it on different subsets of data to ensure it generalizes well. In K-fold cross-validation, the data is split into $k$ subsets where the model is trained on $k-1$ folds and tested on the remaining fold and repeating this process $k$ times [5].

For each iteration, the model is trained on the training set, and parameters like $\beta$ and $\sigma^2$ are estimated. The model is then tested on the test set using metrics like MSE and $R^2$-score. This process is repeated for each fold, and the results are averaged to evaluate the overall performance.

Cross-validation helps prevent overfitting by testing the model on different subsets, providing a more accurate estimate of how it will perform on new, unseen data.

## 3 Implementation

### 3.1 Franke Function

The Franke Function, defined as a sum of four exponentials in Equation 27, is commonly used to benchmark numerical methods in interpolation, regression, and surface fitting. It is particularly effective for testing approximation methods on scattered two-dimensional data.

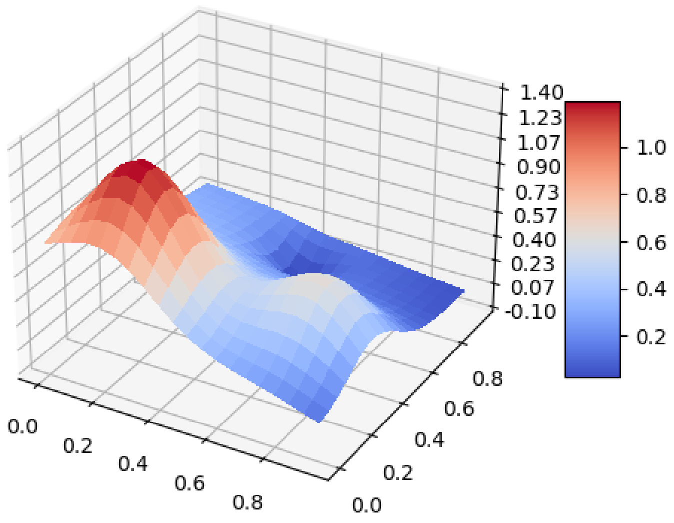


Figure 1: Franke Function visualized.

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \quad (27)$$

In the implementation of OLS, Ridge, Lasso, and Bias-Variance trade-off we initially used 40,000 data points with added stochastic noise. However, due to computational limitations, the dataset was reduced to 100 data points in Cross-validation. For Ridge and Lasso, we evaluated the models across a range of $\lambda = [0.01, 0.1, 1, 10, 100]$. For Cross validation, we were evaluating K for a range of 5-10 folds.

Prior to scaling the data, we randomly split the data into training and test sets. 20% of the data was used for testing, while 80% was used for training. The splitting was occured using Train_test_split function in Scikit_learn [1]. Splitting data into train and test sets allows the model to learn patterns from training set while evaluating its performance on unseen data through the test set. This ensures that the model generalizes well and helps prevent overfitting.

To scale our data we used StandardScaler [1]. The StandardScaler class from scikit-learn is a preprocessing tool that standardizes features by removing the mean and scaling to unit variance [1]. This ensures that all features contribute equally to the model, and improves models performance, especially when scales differ, preventing bias towards larger values.

To avoid numerical instability associated with matrix inversion when performing OLS, we implemented the OLS regression for the Franke Function using the LinerRegression class from Scikit-learn. Instead of manually computing coefficients $\beta$ through matrix operations, we used the fit() method of LinearRegression. The coefficients $\beta$ were estimated using coef_ attribute, which provides a direct way to access the fitted parameter of the model after the model training. Similarly, for Ridge and Lasso, we utilized the Ridge and Lasso classes, instead of performing the matrix inversion [1].

## 3.2 Terrain Data

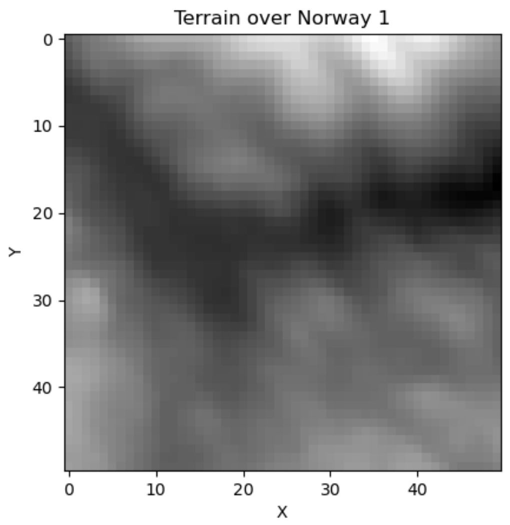


Figure 2: Terrain data

The Terrain data, which was retrieved from Project 1, was sourced from Norway and loaded using imread function. Due to computational complexities, the data was sliced into smaller region for analysis purposes; specifically the range[1000:1050, 500:550]. This resulted in a subset of 2500 data points, representing 50 rows and 50 columns. The regression techniques were applied and split into training and test sets, following an 80/20 split. To ensure all features contributed equally to the model, we standardized the data using the StandardScaler class from Scikit-learn. Given the computational complexity, we evaluated the models across a range of regularization parameters, $\lambda = [0.01, 0.1, 1, 10, 100]$, performed k-fold cross validation with k = 10, and considered polynomial degrees up to a maximum of 10.

## 4 Results

### 4.1 Franke Function

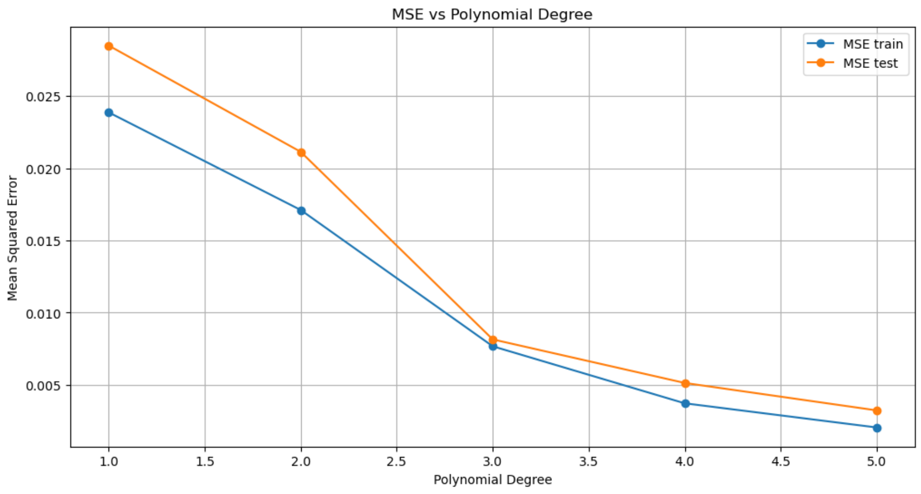


Figure 3: MSE vs Polynomial Degree

Figure 3 shows the MSE for both training and test data as a function of polynomial degree. We can clearly see that as the polynomial degree increases, both in the training and test data, the MSE value decreases. This indicates a improved model performance, since we want the MSE as close to zero as possible. The similarity between the MSE values for the training and test data suggests that the model is able to generalize well to new, unseen data, maintaining accuracy without overfitting to the training set.

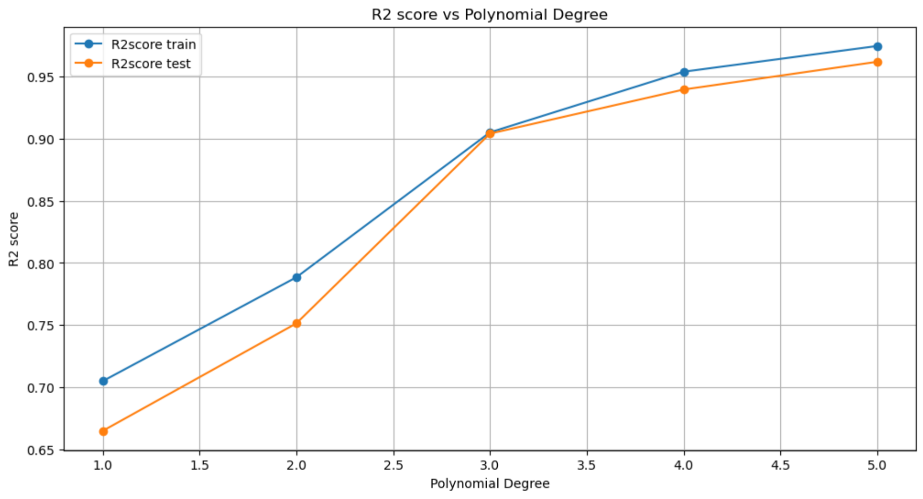


Figure 4: $R^2$ Score vs Polynomial

Figure 4 shows the $R^2$-score for both the training and test data as a function of polynomial degree. The $R^2$-score measures how well the model explains the variance in the data, where values close to 1 indicating a better fit. As the polynomial degree increases, the $R^2$-score improves for both training and test data, showing that the model is better to capture patterns in the data with higher degrees. The training curve is slightly higher than the test curve, which suggests that the model fits the training data slightly better. However, since the test curve also improves, it indicates that the model generalizes reasonably well without servere overfitting up to the fifth degree.

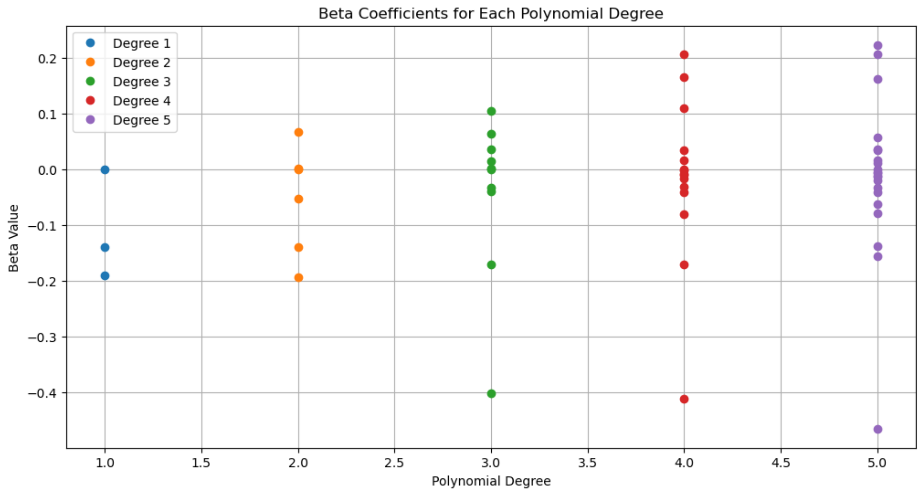


Figure 5: *beta* Coefficients for Each Polynomial Degree

Figure 5 shows the $\beta$ coefficients for polynomial degrees 1-5. Each degree introduces a different number of coefficients, where the higher degree results in more terms. As the degree of the polynomial increases, the model becomes more complex, and the way it fits the data changes. These changes in the $\beta$ values follow a specific trends that are characteristic of different polynomial orders. The number of $\beta$ coefficients increases with the complexity of the polynomial.

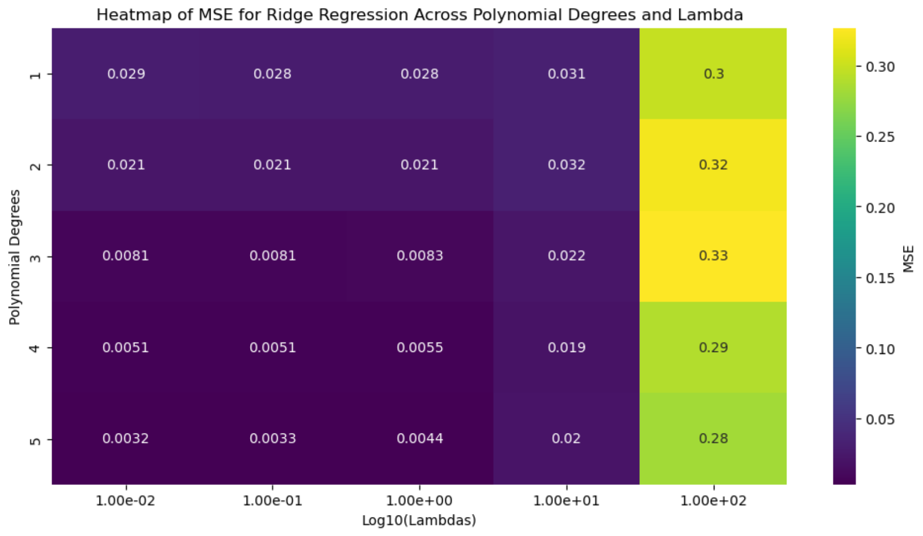


Figure 6: Heatmap of MSE for Ridge Regression Across Polynomial Degrees and $\lambda$

Figure 6 displays the MSE for Ridge Regression for the polynomial degrees and $\lambda$. As the polynomial degree increases, the values MSE goes closer toward zero, which reflects better model performance. The MSE remains low for smaller $\lambda$ values. However, as $\lambda$ increases, the MSE also rises significantly, showing that high regularization parameter reduces the models predictive accuracy in this specific data.

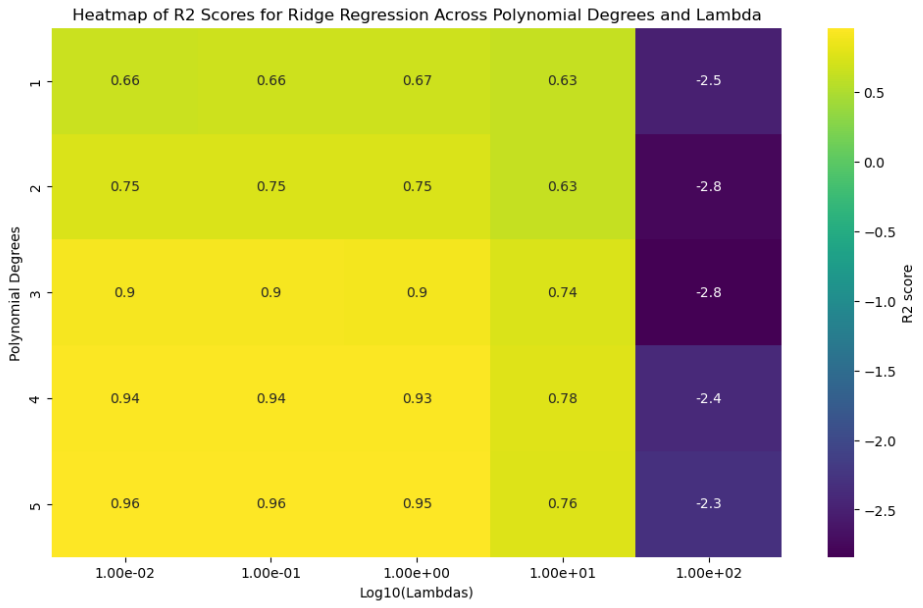


Figure 7: Heatmap of $R^2$-scores for Ridge Regression Across Polynomial Degrees and $\lambda$

Figure 7, shows that as the polynomial degree increases, the $R^2$-scores generally improve, indicating a better fit. The smaller the $\lambda$ values are, the $R^2$-scores are closer to 1. However, as $\lambda$ increases, especially for very large values ($\lambda > 10$), the $R^2$-score become negative, suggesting that the model's predictions are less accurate than just using the average of the actual values.

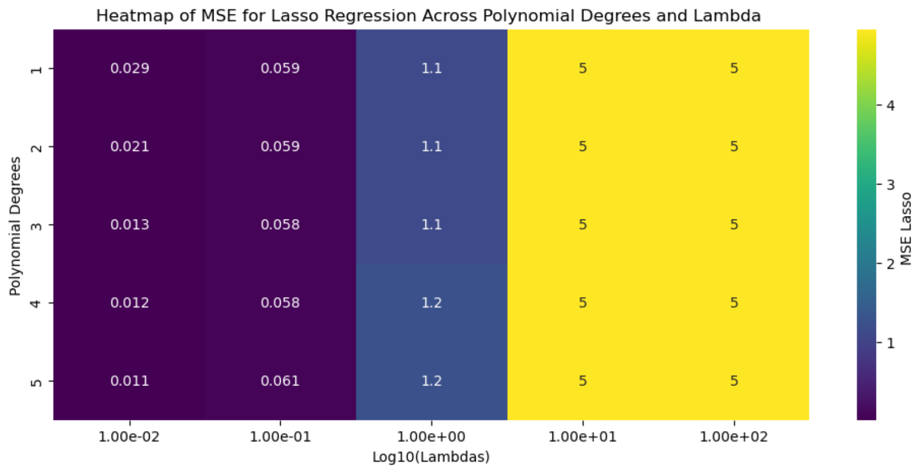


Figure 8: Heatmap of MSE for Lasso Regression Across Polynomial Degrees and $\lambda$

Figure 8 shows the MSE values for Lasso regression. Lower MSE values occur with lower $\lambda$ values and higher polynomial degrees, while larger $\lambda$ values lead to higher MSE, indicating decreased model performance.

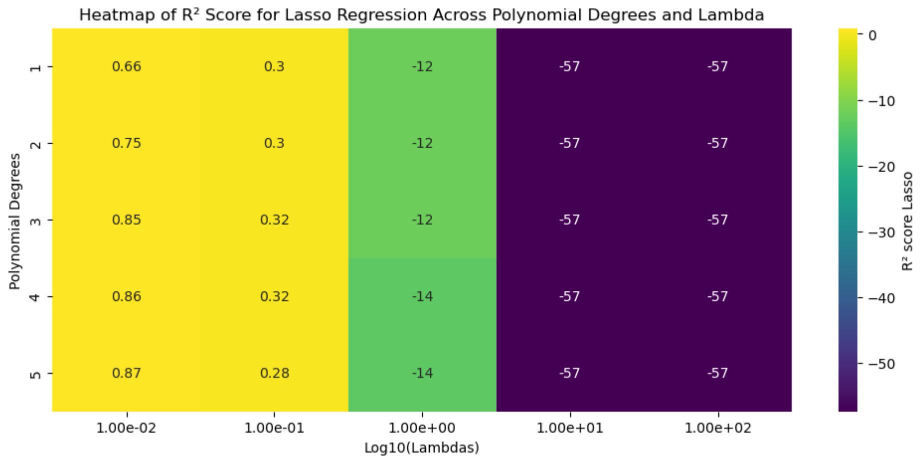


Figure 9: Heatmap of $R^2$-scores for Lasso Regression Across Polynomial Degrees and $\lambda$

The heatmap 9 shows the $R^2$-scores for Lasso regression. Similar to Ridge in Figure 7, higher polynomial degrees result in better $R^2$-scores, while larger $\lambda$ values significantly reduce the $R^2$-scores, particularly when $\lambda$ exceeds 1.

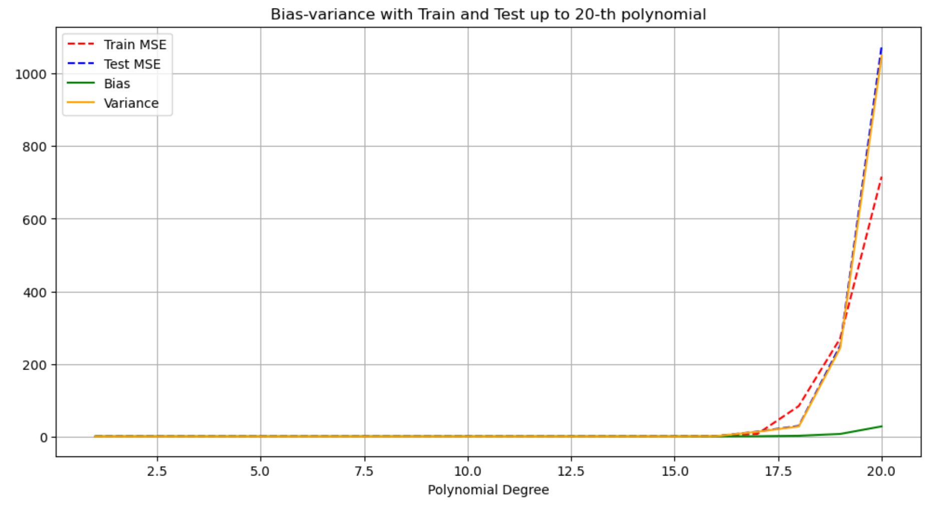


Figure 10: Bias-Variance with Train and Test MSE up until the 20th Polynomial

The plot 10 shows the train MSE, test MSE, Bias and Variance as functions of the polynomial degree, with the model complexity reaching up to 20-th polynomial. Both the train and test data for MSE remains relatively low and stable for most of the polynomial degrees, up until around the 16-18th polynomial degree. The sudden rise in both Train and espically Test reflects the models tendency to overfit, capturing noise rather than the underlying patterns, leading to poor generalization. And bias remains low throughout, while the variance suddenly increases around polynomial degree 18. With the low bias, high variance situation means that the model overfits the training data and performs poorly when we apply it to our test data, as we can clearly see in Figure 10.

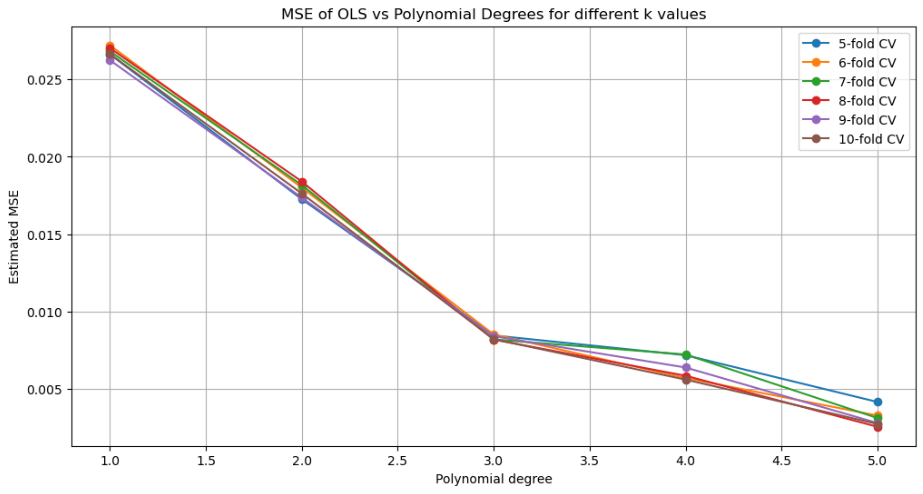


Figure 11: MSE of OLS vs Polynomial Degrees for Different k Values

The plot in Figure 11 shows the MSE for OLS for different cross-validation folds ($k = 5, 6, 7, 8, 9, 10$). As the polynomial degree increases, the MSE decreases for all cross-validation folds. This indicates a stable model performance across various cross-validation patterns.

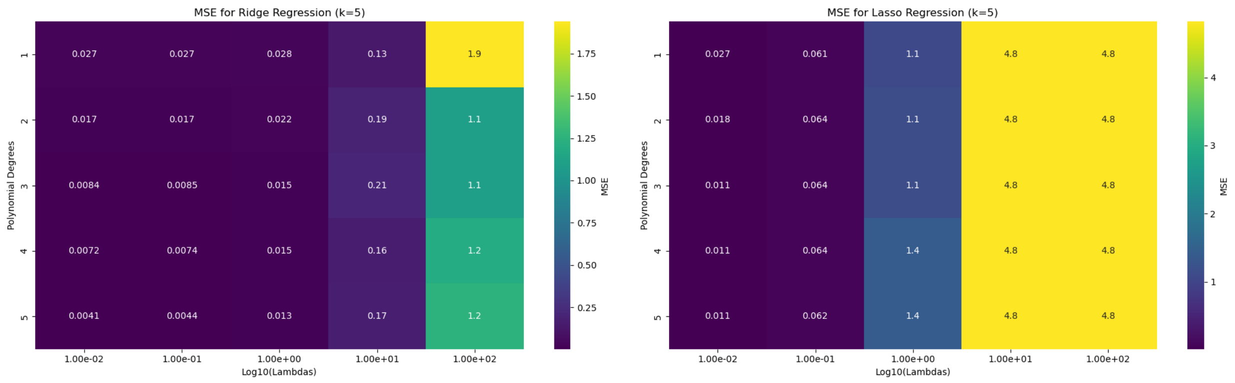


Figure 12: MSE for Ridge and Lasso using 5-fold Cross Validation

Figure 12 shows a heatmap of MSE values for both Ridge and Lasso Regression using 5-fold Cross Validation. For MSE Ridge Regression, we can see that the value decreases as polynomial degrees increase and remains relatively low for smaller $\lambda$ values. However, as $\lambda$ increases, the MSE also rises. This is similar to Figure 6 but the MSE values for without any cross validation were better than the values for 5-fold cross validation. Similarly for Lasso Regression, MSE stays low for smaller $\lambda$ values but rises a lot for larger $\lambda$ values, particularly for $\lambda \geq 1$. Compared to Figure 8, the MSE values for Lasso are quite the same for the lowest $\lambda$ value but 5-fold cross-validation is better compared to the higher $\lambda$ values.

Both Ridge and Lasso regression show lower MSE at smaller $\lambda$ values and higher polynomial degrees. However, Lasso has larger MSE values compared to Ridge overall, which makes Ridge a better model than Lasso for 5-fold cross validation.

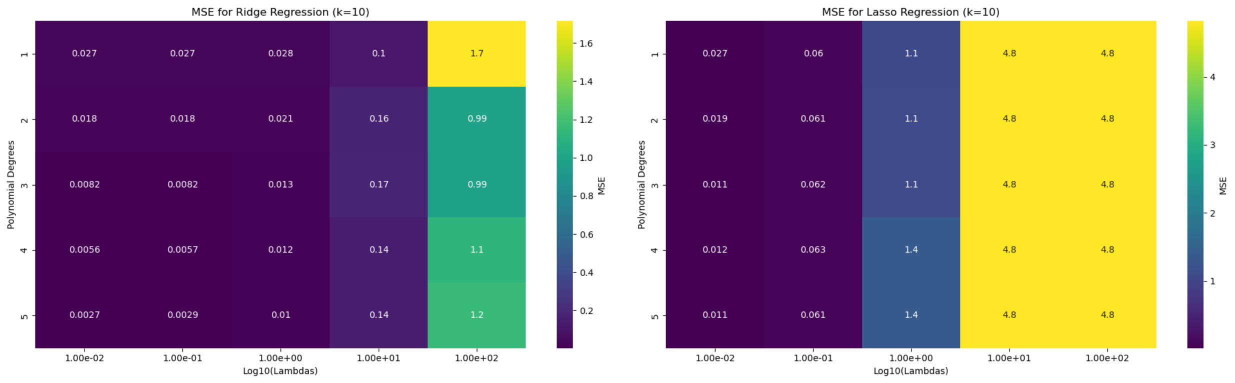


Figure 13: MSE of OLS using 10-fold Cross Validation across Polynomial Degree for Terrain data.

Figure 13 shows a heatmap of MSE values for both Ridge and Lasso Regression using 10-fold Cross Validation. As all the previous figures the MSE decreases with higher polynomial degrees and lower $\lambda$ values, but increases significantly for larger $\lambda$ values. Compared to Figure 6 and 8, 10-fold cross validation gives better values for the low $\lambda$ values. And also here, Ridge has better MSE values than Lasso, meaning it is a better model for our case.

## 4.2 Terrain Data

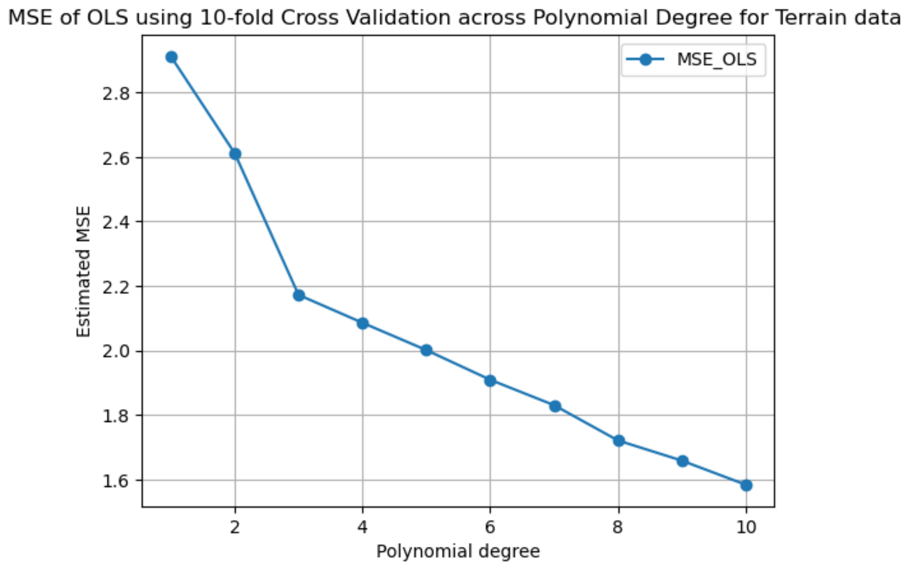
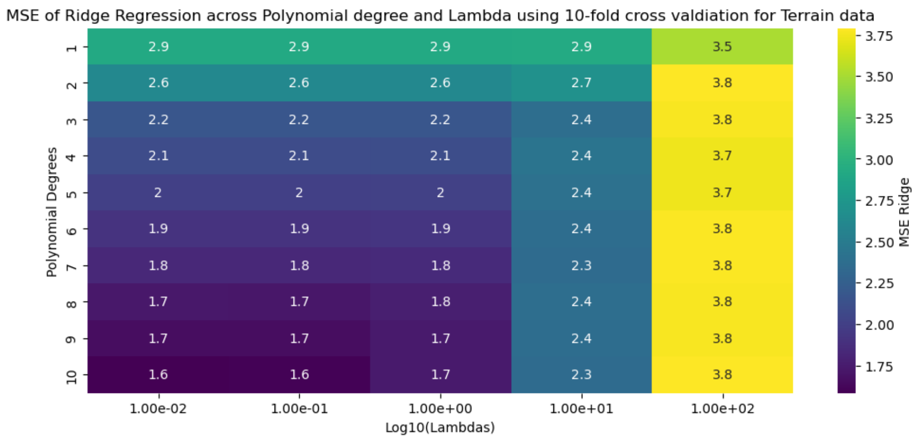


Figure 14: MSE of OLS vs Polynomial Degrees for Terrain Data

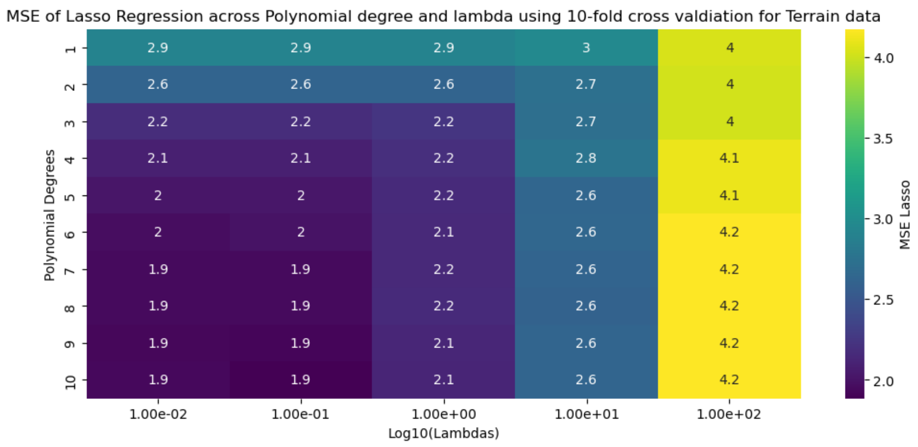
Figure 14 illustrates the MSE for Ordinary Least Squares (OLS) regression across polynomial degrees (1 to 10) on the terrain dataset, using 10-fold cross validation. As the polynomial increases, the MSE steadily decreases, indicating better model performance with more complex features.The MSE drops from around 2.8 at degree 1 to around 1.6 at degree 10, reflecting improved fitting with higher-degree polynomial.



| Polynomial Degrees | 1.00e-02 | 1.00e-01 | 1.00e+00 | 1.00e+01 | 1.00e+02 |
|---|---|---|---|---|---|
| 1 | 2.9 | 2.9 | 2.9 | 2.9 | 3.5 |
| 2 | 2.6 | 2.6 | 2.6 | 2.7 | 3.8 |
| 3 | 2.2 | 2.2 | 2.2 | 2.4 | 3.8 |
| 4 | 2.1 | 2.1 | 2.1 | 2.4 | 3.7 |
| 5 | 2 | 2 | 2 | 2.4 | 3.7 |
| 6 | 1.9 | 1.9 | 1.9 | 2.4 | 3.8 |
| 7 | 1.8 | 1.8 | 1.8 | 2.3 | 3.8 |
| 8 | 1.7 | 1.7 | 1.8 | 2.4 | 3.8 |
| 9 | 1.7 | 1.7 | 1.7 | 2.4 | 3.8 |
| 10 | 1.6 | 1.6 | 1.7 | 2.3 | 3.8 |

Figure 15: MSE of Ridge Regression across Polynomial degree and Lambda using 10-fold cross valdiation for Terrain data.

Figure 15 illustrates a heatmap of the Mean Squared Error (MSE) for Ridge regression across different polynomial degrees and $\lambda$ values for cross-validation on the terrain dataset. As the polynomial degree increases, the MSE decreases, particularly for higher degrees such as 9 and 10. The effect $\lambda$ is more significant for larger values, where MSE increases at higher $\lambda$ values, particularly beyond $\lambda = 100$.



| Polynomial Degrees | 1.00e-02 | 1.00e-01 | 1.00e+00 | 1.00e+01 | 1.00e+02 |
|---|---|---|---|---|---|
| 1 | 2.9 | 2.9 | 2.9 | 3 | 4 |
| 2 | 2.6 | 2.6 | 2.6 | 2.7 | 4 |
| 3 | 2.2 | 2.2 | 2.2 | 2.7 | 4 |
| 4 | 2.1 | 2.1 | 2.2 | 2.8 | 4.1 |
| 5 | 2 | 2 | 2.2 | 2.6 | 4.1 |
| 6 | 2 | 2 | 2.1 | 2.6 | 4.2 |
| 7 | 1.9 | 1.9 | 2.2 | 2.6 | 4.2 |
| 8 | 1.9 | 1.9 | 2.2 | 2.6 | 4.2 |
| 9 | 1.9 | 1.9 | 2.1 | 2.6 | 4.2 |
| 10 | 1.9 | 1.9 | 2.1 | 2.6 | 4.2 |

Figure 16: MSE of Lasso Regression across Polynomial degree and lambda using 10-fold cross valdiation for Terrain data.

Figure 16 presents a heatmap of MSE for Lasso regression across various polynomial degrees and $\lambda$ values with cross-validation on the terrain dataset. Similar to Ridge, the MSE decreases with higher polynomial degrees, especially at degree 10. However, Lasso shows more sensitivity to higher $\lambda$ values, where MSE increases significantly at $\lambda > 10$, indicating that strong regularization in Lasso reduces the complexity of the model.

## 5 Discussion

### 5.1 Comparison OLS, Ridge and Lasso

What differenciates Ridge 20 and Lasso Regression 24 from the OLS 6, is the regularization parameter $\lambda$ . Ridge and Lasso regression both apply the regularization parameter $\lambda$, which controls the penalty on the models coefficients, but they handle this penalty differently.

In Ridge regression, $\lambda$ penalizes the squared values of the coefficients, shrinking them uniformly toward zero as $\lambda$ increases, which helps reduce overfitting without eliminating features entirely. In contrast, Lasso regression penalizes the absolute values of the coefficients, allowing it not only to shrink them but also to force some coefficients to exactly zero, which helps feature selection by eliminating less important variables. In both methods, choosing the right $\lambda$ is important. If it's too small, the model will act like OLS, while if it's too large, the model shrinks the coefficients and can miss important patterns. Therefore, $\lambda$ must be carefully tuned to balance bias and variance.

To evaluate which model fits the data best, we compare the performance of OLS, Ridge and Lasso based on their MSE and $R^2$-scores across different polynomial degrees and $\lambda$ values. OLS shows improvement in both MSE and $R^2$-score as the polynomial degree increases, especially for higher-degree polynomials. However, without regularization, OLS is likely to overfitting, which can lead to poor generalization on unseen data, even though it achieves low error on the training set. Ridge and Lasso, on the other hand, use regularization to address overfitting. Both Ridge and Lasso have good performance at lower $\lambda$ values, reducing MSE and improving $R^2$-scores, especially with high-degree polynomials.

When comparing all three models: OLS, Ridge and Lasso, OLS and Ridge outperforms Lasso. This may be because Lassos regularization term forces some coefficients to exactly zero potentially leading to worse results on this particular data. Both Ridge and OLS perform well, but OLS appears to have a slight better values for this particular data. As shown in Figures 3 and 4, OLS achieves an MSE $\approx 0.0025$ and an $R^2$-score $\approx 0.975$ for the training data. In comparison, Ridge, as seen in Figures 6 and 7, has an MSE of 0.0032 and an $R^2$-score of 0.96. While OLS performs marginally better on this specific dataset, Ridge would likely offer the best fit in the long run due to its ability to handle multicollinearity and reduce the influence of less important features without completely eliminating them.

## 5.2 Bias Variance and Resampling

The bias-variance trade-off becomes important when comparing the two methods of model evaluation: bootstrap resampling with a large dataset (n = 40,000) in Figure 10 and k-fold cross-validation (n = 100) in Figure 11. In the bootstrap resampling analysis, as the polynomial degree increases over 16, the test MSE rises dramatically due to high variance, indicating severe overfitting. While higher polynomial degrees reduce bias, they also increase variance significantly, leading to poor generalization to unseen data. This is clearly reflected in the large gap between training and test MSE at high degrees in Figure 10, where the model fits the training data well but struggles on new data.

In contrast, the k-fold cross validation results, which involve lower polynomial degrees (up to 5), shows a more controlled and steady decrease in MSE as model complexity increases. With fewer data points and lower polynomial complexity, the risk of overfitting is minimized, and the MSE consistently decreases across all k-values. This comparison highlights how model complexity affects bias and variance differently depending on the size of the dataset. With larger datasets and more complex models, regularization techniques like Ridge and Lasso become crucial for managing the variance while still capturing the underlying patterns in the data.

## 5.3 Terrain Data

In this analysis, we assessed the performance of Ridge, OLS and Lasso models on a dataset consisting of 2500 terrain data points, employing 10-fold cross-validation and polynomial degrees ranging from 1 to 10. We used the 10-fold cross-validation method to test the model on different subsets of the data, ensuring that the results are reliable. Both Ridge and Lasso employ regularization through the parameter $\lambda$ to control the complexity of the model and preventing overfitting, while OLS, lacking regularization, is more vulnerable to overfitting as the polynomial degree increases.

OLS shows a steady reduction in Mean Squared Error (MSE) as the polynomial degree increases, reaching its best performance at degree 10, where the MSE $\approx 1.6$, shown in Figure 14. This suggests that OLS can capture the complexity of the terrain data when high-order polynomial terms are included. However, the lack of regularization is a significant drawback, as higher-degree polynomials can lead to overfitting, especially with a model like OLS that does not have a mechanism to penalize complex models. While the MSE decreases with increasing polynomial degree, OLS may not generalize well to new data due to its tendency to overfit.

Ridge Regression, on the other hand, addresses the overfitting issue through regularization, which penalize large coefficients. The heatmap of Ridge shows that the MSE also decreases with increasing polynomial degree, reaching its lowest values around 1.6 at degrees 9 and 10, particularly when $\lambda$ is in the range $10^{-2}$ to $10^{-1}$, which we can clearly see in Figure 15. This indicates that Ridge effectively captures the complexity of the terrain data without overfitting, due to regularization that prevents the model from becoming complex. Ridge achieves a balance between flexibility (by using higher-degree polynomials) and generalization (through regularization), making it a robust model for this data.

Lasso Regression in Figure 16 performs less effectively compared to Ridge and OLS. While Lasso benefits from higher polynomial degrees, its regularization strategy is more aggressive, shrinking some coefficients to zero, which results in elimination of important features. This tendency is particularly pronounced as $\lambda$ increases, leading to significant rise in MSE (above 4.0 at higher $\lambda$), indicating to underfitting. The strong regularization of Lasso may eliminate higher-polynomial terms that are necessary for accurately modeling the terrain data, which limits its performance, especially at higher levels of $\lambda$.

To conclude the best fitting model, Ridge Regression emerges as the best overall model for this dataset, offering an optimal balance between complexity and regularization. While OLS fits the data well at higher polynomial degrees, its risks overfitting without regularization. The tendency of Lasso to eliminate important features limits its ability to model terrain data effectively. Rigde, with its controlled regularization and ability to generalize across different data splits, proves to be the most reliable and the most suitable model for this complex terrain dataset.

## 6 Conclusion

When comparing OLS, Ridge, and Lasso regression models, it becomes clear that regularization is important for managing model complexity and preventing overfitting. OLS does well on training data, especially with higher polynomial degrees, but without regularization, it tends to overfit, meaning it performs poorly on new, unseen data. Ridge and Lasso, by using a regularization parameter $\lambda$, help prevent overfitting by reducing the size of the coefficients. Ridge is particularly effective because it reduces overfitting without completely removing features, while Lasso can be too strict by forcing some coefficients to zero, which can lead to underfitting, especially with complex datasets like terrain data.

From the analysis, Ridge Regression consistently outperforms both OLS and Lasso, especially when the regularization parameter $\lambda$ is low. It manages to capture the complex patterns in the terrain data without overfitting, making it the most reliable model overall. OLS may offer slightly better performance in terms of MSE on this specific dataset, but its vulnerability to overfitting makes it a less robust choice in practice. Lasso can eliminate important features and it limits its effectiveness for this dataset. In conclusion, Ridge regression is the best-fitting model because it balances flexibility and generalization well. This makes it a strong choice for complex data like terrain models, as it can capture patterns without overfitting.

## 7 Appendix

Link to Github: `https://github.uio.no/arangans/Project_1.git`

In this GitHub link, you will find the entire code of this project, plots, image and heatmaps that have and not been included in our report. This link do also include the terrain data that has been used for the analysis of real data.

## 8 Acknowledgement

During this project, we gathered essential information for both the code and theoretical concepts from Professor Morten Hjorth-Jensen's GitHub page.

The ChatGPT tool has been valuable a lot during this project. It has been used to:

- Assist with adapting and modifying the code from weekly exercises, focusing in both the Franke function and real-world data analysis, particulary in generating plots.
- Review and refined the grammar, enhancing the overall clarity and language of the project.
- Used for to code the math equations, and helped us a lot with overleaf in general.

## 9 References

## References

[1] Lars Buitinck et al. *API design for machine learning software: experiences from the scikit-learn project.* 2013. arXiv: `1309.0238 [cs.LG]`. URL: `https://arxiv.org/abs/1309.0238`.

[2] Wikipedia contributors. *Ordinary Least Square.* Accessed: 2024-09-30. URL: `https://en.wikipedia.org/wiki/Ordinary_least_squares`.

[3] Morten Hjorth-Jensen. *3. Linear Regression.* Accessed: 2024-10-07. 2021. URL: `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter1.html#introduction`.

[4] Morten Hjorth-Jensen. *4. Ridge and Lasso Regression.* Accessed: 2024-10-07. 2021. URL: `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter2.html`.

[5] Morten Hjorth-Jensen. *5. Resampling Methods.* Accessed: 2024-10-07. 2021. URL: `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter3.html`.

[6] Morten Hjorth-Jensen. *Week 35: From Ordinary Linear Regression to Ridge and Lasso Regression.* Accessed: 2024-10-07. 2024. URL: `https://github.com/CompPhysics/MachineLearning/blob/master/doc/pub/week35/ipynb/week35.ipynb`.

[7] IBM. *What is Linear Regression?* Accessed: 2024-10-07. URL: `https://www.ibm.com/topics/linear-regression`.

[8] Trist'n Joseph. *What Is Bootstrapping Statistics?* Accessed: 2024-10-07. 2023. URL: `https://builtin.com/data-science/bootstrapping-statistics`.