Regression analysis and Resempling methods
Project 1

Eleonora Gandini Mazzarelli, Anna Milanesi, Mia Margrethe Ramberg Storstad

October 7, 2024

Abstract

In this report, we explored three distinct regression methods for modeling and predicting data: Ordinary Least Squares (OLS), Ridge Regression, and Lasso Regression. We first applied these methods to Franke Function data, followed by an analysis using real terrain data from a region near Stavanger, analysing MSE and R^2 in order to find the best fitting model. We then conducted a detailed data analysis using two resample techniques: bootstrap for the OLS regression, and k-fold cross validation. Through this analysis we discovered that the best fitting model for the Franke Function was provided by Ridge Regression, while for the terrain data the OLS model performed better.

Click here to go the GitHub repository

Contents

1 Introduction

Linear regression models, while simple, are a useful tool for predicting a quantitative response and serves as a strong foundation for more advanced approaches. Many modern statistical learning techniques can be viewed as generalizations or extensions of linear regression.
Therefore, this project investigates the use of Lasso, Ridge, and Ordinary Least Squares (OLS) regression as three important regression techniques for predictive modeling. We are going to test these methods on the Franke function and real-terrain data to determine which one is best suited, with an emphasis on model evaluation using metrics like the Mean Squared Error (MSE) and R^2 score. To improve the model's predictive performance we will concentrate on the bias-variance trade-off and use resampling approaches, like bootstrap and k-fold cross validation. We will look into model complexity, variability, and distribution properties in order to reach the ultimate goal of this work: provide guidance for fitting linear regression models to datasets [1].
The project consist of 4 main sections: in section 2 we will explain the methods used and derive relevant equations. A general introduction to the data studied is presented in section 3. In section 4 we will provide results and discuss performances and consequences, and lastly, our conclusion will be presented in section 5.

2 Method

2.1 Linear Regression

A linear regression model tries to describe the relationship between input variables $\mathbf{x}^T = (x_0, x_1, \ldots, x_{p-1})$ and output variable y by assuming a linear dependency. The model also incorporates random noise normal distributed $\epsilon \sim N(0, \sigma^2)$.

$$y = f(x) + \epsilon$$

We approximate the function $f(x)$ with our model $\tilde{y}$

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

For an observed value y_i, we have

$$y_i = \Sigma_{j=0} x_{ij}\beta_j + \epsilon_i = \mathbf{X}_{\mathbf{i},*}\beta + \epsilon$$

The product $\mathbf{X}_{\mathbf{i},*}\beta$ is deterministic, and ϵ is normalized, so when taking the expectation we get

$$\mathbb{E}(y_i) = \mathbb{E}(\mathbf{X}_{\mathbf{i},*}\beta) + \mathbb{E}(\epsilon_i) = \mathbf{X}_{\mathbf{i},*}\beta + 0$$

$$\mathbb{E}(y_i) = \mathbf{X}_{\mathbf{i},*}\beta$$

Its variance is given by

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

The aim is to determine the best-fitting line by minimizing a cost function, using methods such as Ordinary Least Squares (OLS), Ridge, and Lasso regression. The main challenge is to balance model complexity, using the bias-variance trade-off to avoid overfitting or underfitting the data.

2.1.1 Ordinary Least Square

The OLS cost function:

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

when minimized gives us the OLS expression for the optimal parameters $\hat{\beta}$

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

Taking the expectation for the optimal parameters we get

$$\begin{aligned}
\mathbb{E}(\hat{\beta}_{OLS}) &= \mathbb{E}((\mathbf{X^T X})^{-1}\mathbf{X^T y}) \\
&= (\mathbf{X^T X})^{-1}\mathbf{X^T}\mathbb{E}(\mathbf{y}) \\
&= (\mathbf{X^T X})^{-1}\mathbf{X^T}\beta \\
&= \beta
\end{aligned}$$

With the variance:

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

We refer to The Elements of Statistical Learning [2], chapter 3, for assistance with this demonstration.

2.1.2 Ridge Regression

Ridge Regression is an alternative to OLS, but with a added term for regulating overfitting the model [3]. The Ridge Regression cost function is simply the OLS cost function with the added regularization term $\lambda\beta^T\beta$:

$$C_{Ridge}(\beta) = \frac{1}{n}((\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)) + \lambda\beta^T\beta$$

$\beta^T\beta$ is the sum of the squared values of all coefficients: $\beta^T\beta = \Sigma_{j=0}^k\beta_j^2$ [4], and λ is the regularization parameter that controls the importance of the regularization term. If λ is higher, it regulates and shrinks the coefficients to prevent overfitting.
To find the optimal parameters for Ridge regression, we take the derivative with respect to β, set it to zero and solve for β. The solution gives the optimal $\hat{\beta}_{Ridge}$:

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

2.1.3 Lasso Regression

The Lasso regression cost function

$$C_{LASSO}(\beta) = \frac{1}{n}((\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)) + \lambda||\beta||_1$$

is, as the Ridge cost function, based on the OLS cost function but with an added regularization term $\lambda||\beta||_1 = \lambda\Sigma_{j=0}^k|\beta_j|$. This regularization term penalizes the absolute value of the coefficients, which allows some coefficients to shrink to zero [4].
For Lasso regression, it is not possible to find the optimal β as we have done for Ridge and OLS, because the absolute values in the regularization term is not differentiable at zero [4].

2.2 MSE and R^2

The Mean Squared Error (MSE) calculates the average of squared errors, i.e., average squared difference between the expected and actual values. A sample of n data points' MSE is calculated as follows:

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

MSE is commonly employed because squaring the differences reduces the weight of tiny deviations while significantly penalizing big ones.

The coefficient of determination R^2 is another statistic used to assess model fit:

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

Where $\tilde{y}_i$ is the predicted i-th value of the sample and y_i is the corresponding true value. The coefficient of determination allows to see how effectively the model predicts future samples. In the regression context, it is particularly useful for determining how well the regression line approximates the real data.
Increasing the complexity of a linear model (e.g., by increasing the polynomial degree), the MSE tends to reach 0, and the R^2 approaches 1. However when the model overfits, it becomes inaccurate when confronted with new data. The aim is to determine the ideal model complexity that maximizes R^2 and minimizes MSE in the test data.
In most cases, the data is divided into two sets: a training set and a test set. Occasionally, a validation set is also utilized, although not in this instance. The test set assesses the model's accuracy, whereas the training set is used for building it. A typical strategy is to use 80% of the data for training and 20% for testing, even though there is no set rule for the split ratio [5].

2.3 Scaling data

Feature scaling is an important step in machine learning preprocessing because real-world data frequently contains features with variable units and sizes. This variance has a substantial influence on models such as Mean Squared Error (MSE), which are sensitive to data scale. To remedy this, scaling characteristics makes them more similar and prevents biased findings owing to outliers. The most commonly used scaling technique is Standardization, which adjusts the characteristics by subtracting the mean and dividing by the standard deviation. This produces features with a mean of zero and a standard deviation of one. For each feature:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \bar{x}_j}{\sigma(x_j)}$$

where $\bar{x}_j$ and $\sigma(x_j)$ are the mean and standard deviation, respectively, of feature x_j.

2.4 Bias-Variance

The parameters β are in turn found by optimizing the mean squared error via the so-called cost function:

$$C(x, \beta) = \frac{1}{n}\sum_{j=1}^{n}(\hat{y}_j - y_j)^2 = \mathbb{E}[(\hat{y} - y)^2]$$

This is the expected value of the MSE, that can be written on the form:

$$\text{MSE}(y, \tilde{y}) = \mathbb{E}[(y - \hat{y})^2] = \text{Bias}[\hat{y}] + \text{Var}[\hat{y}] + \sigma^2$$

where:

$$\text{Bias}[\hat{y}] = \mathbb{E}[y] - \mathbb{E}[\hat{y}], \quad \text{Var}[\hat{y}] = \mathbb{E}[(\hat{y} - \mathbb{E}[\hat{y}])^2]$$

to derive this expression, first of all we can write $y = f + \epsilon$, where f is a non-stochastic variable and ϵ is a stochastic variable and its expected value is zero.
Then we can start by decomposing the MSE general expression and do some calculations:

$$\mathbb{E}[(y - \hat{y})^2] = \mathbb{E}[y^2 - 2y\hat{y} + \hat{y}^2] = \mathbb{E}[y^2] - 2\mathbb{E}[y\hat{y}] + \mathbb{E}[\hat{y}^2]$$

We can now rewrite the three terms of the expression above:

- $\mathbb{E}[\hat{y}^2] = \text{Var}[\hat{y}] + (\mathbb{E}[\hat{y}])^2$
- $\mathbb{E}[y^2] = \mathbb{E}[(f + \epsilon)^2] = \mathbb{E}[f^2] + 2\mathbb{E}[f\epsilon] + \mathbb{E}[\epsilon^2] = f^2 + \sigma^2$
  where we used that f^2, like f, is a non-stocastic variable, and therefore its expected value is f^2 itself and that the expected value of ϵ can be rewritten as σ^2. We also used that, since f and ϵ are independent, $\mathbb{E}[f\epsilon] = \mathbb{E}[f]\mathbb{E}[\epsilon] = 0$,
- $\mathbb{E}[y\hat{y}] = \mathbb{E}[(f + \epsilon)\hat{y}] = \mathbb{E}[f\hat{y}] + \mathbb{E}[\epsilon\hat{y}] = \mathbb{E}[f\hat{y}] + \mathbb{E}[\epsilon]\mathbb{E}[\hat{y}] = f\mathbb{E}[\hat{y}]$

Collecting all the expressions we obtain:

$$\mathbb{E}[(y - \hat{y})^2] = f^2 - 2f\mathbb{E}[\hat{y}] + (\mathbb{E}[\hat{y}])^2 + \sigma^2 = \mathbb{E}[f^2] - 2\mathbb{E}[f\hat{y}] + (\mathbb{E}[\hat{y}])^2 + \text{Var}[\hat{y}] + \sigma^2$$

We can then rewrite $(f - \mathbb{E}[\hat{y}])^2 \approx (y - \mathbb{E}[\hat{y}])^2$ and obtain the final expression.
The bias, which is the discrepancy between the model's expectation value and true function, indicates how well a model captures data complexity. The variance represents the discrepancy between the model and the expected value in the model. Low variance indicates stability, however large variance might result in overfitting by catching noise instead of underlying patterns.
Balancing bias and variance is critical since a simple model may be consistent but ignore significant associations, whereas a complicated model may perform well on training data but struggle with new data. Achieving this equilibrium is critical for making accurate predictions about unseen data.

2.5 Resample Techniques

Resampling techniques are used to obtain insights into models by repeatedly pulling samples from a training set. This strategy enhances evaluation and comprehension when compared to using the original data only once. Bootstrapping and K-fold cross validation (CV) are the two most common approaches.

2.5.1 Bootstrap

The Bootstrap technique estimates distribution properties by randomly sampling n elements multiple times. This technique enables the construction of numerous models using slightly different resampled datasets. As the number of bootstraps rises, the sample's statistical features start to match those of the population. The Bootstrap approach provides insights into the credibility of the model's conclusions by averaging characteristics (such as the mean) obtained from the arrays using simulated variations.

2.5.2 K-fold Cross-Validation

K−fold cross validation subdivides the dataset into K subsets. One subset is utilized for testing, while the rest K−1 used for training. This procedure is repeated until each subset is used as test data, resulting in performance measurements such as Mean Squared Error (MSE). The average of these metrics yields an overall performance score.

3 Data

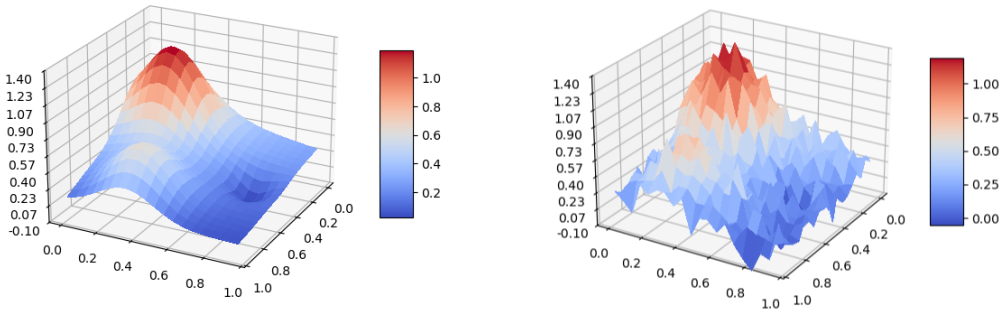
The library numpy [6] was utilized to store and arrange the data, matplotlib [7] was used to create line plots, and seaborn [8] was utilized to create heatmaps. To implement the functions for lasso, k-fold, and bootstrap cross validation, the package scikit-learn [9] was utilized.

3.1 Franke's Function

Franke's bivariate test function is a common two-dimensional test function for interpolation and fitting. It represents the weighted sum of four exponential. It is initialized as follows:

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

where x, y $\in$ [0, 1]. For this reason, we won't scale our Franke's function data, as the input values are already scaled to a particular domain.
We create a dataset with uniformly distributed values of x, y $\in$ [0, 1] and we added a normally distributed noise $\epsilon = \mathrm{N}(0, \sigma^2)$ (see Fig. 1).

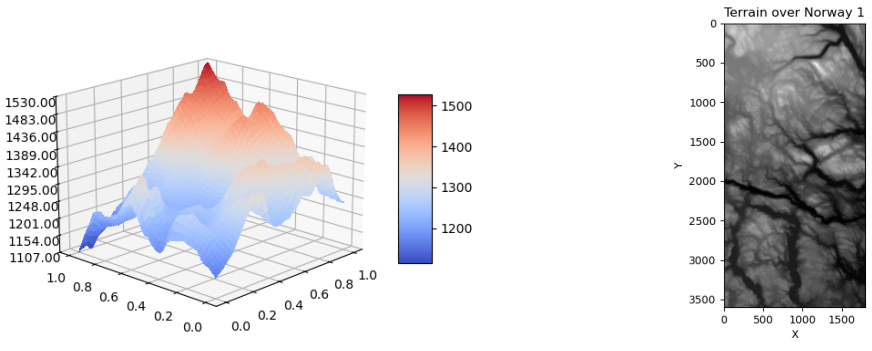


(a) (b)

Figure 1: Franke function without noise (a) and with noise (b).

3.2 Terrain data

After testing the methods on the Franke function, we will test them using real terrain data provided by Morten Hjorth-Jensen. This terrain data is from an area close to Stavanger, and originally had 3601x1801 data points. To simplify the analysis, a subset for the terrain was created with 100x100 data points . An image of the terrain data representing the part of the map is presented in Figure 2.



(a) (b)

Figure 2: Terrain data in 3D (a) and 2D (b) visualized with colors. The color bar represents the height in the terrain.

4 Results and Discussion

4.1 Franke Function

4.1.1 OLS

We used our own code to run a standard ordinary least square regression analysis.

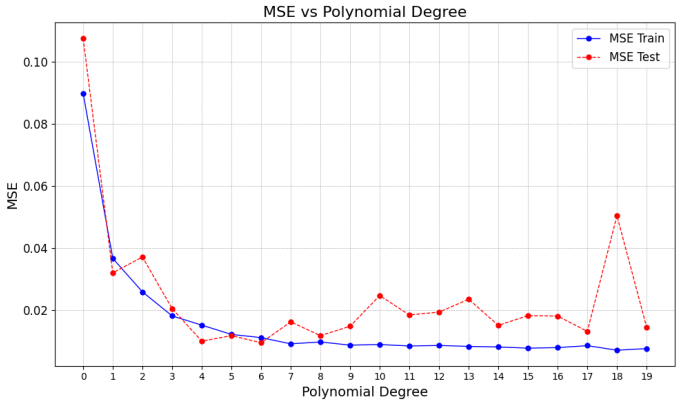


Figure 3: Line plot of MSE vs polynomial degree comparing training and test data in an OLS regression model for 400 data points.

Fig. 3 shows the MSE values for the training and testing data of the Franke function using 400 data points across polynomial degrees from 0 to 19.
The MSE for both training and testing decreases significantly from degree 0 to degree 4. From degree 5 to 17, the MSE for training data continues to decrease and stabilizes, while the MSE for the test data starts fluctuating. A spike in test MSE is observed at polynomial degree 18, followed by a return to lower MSE values for higher degrees. From degree 5 to 19, the test MSE is slightly higher than the train MSE.
Therefore, as model complexity rises, the model undergoes a shift from underfitting from 0 till 3rd degree to overfitting for highest degrees. The peak at degree 18 indicates an overfitting threshold, beyond which the model may re-adjust. This shows how important it is to strike a balance between model complexity and identifying appropriate trends in the data to prevent overfitting.
We plotted the beta values of four distinct polynomial degrees, from the 2th degree till the 5th degree, for the OLS model (Fig. 4).

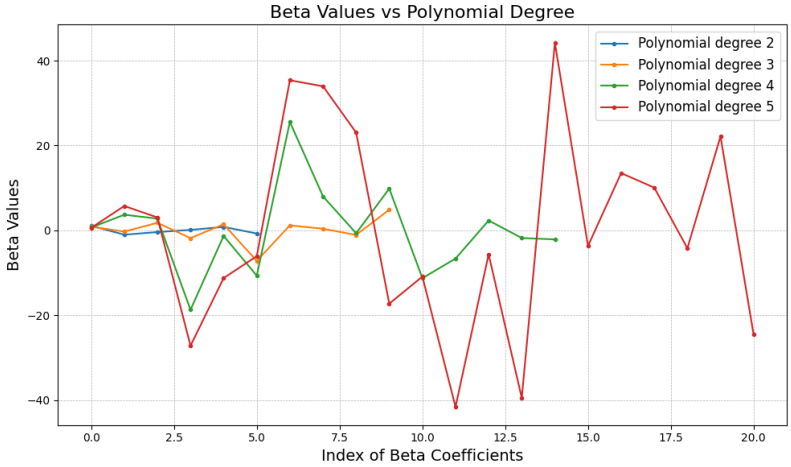


Figure 4: Bar plots showing the coefficients β of polynomial regression models for degrees 2, 3, 4, and 5.

The graph shows the increasing complexity of a polynomial model as its degree increases. The variability of the beta value increases along with the degree of the polynomial.
Fig. 5 shows the MSE values for the training and testing data of the Franke function using 10000 data points across polynomial degrees from 0 to 19.

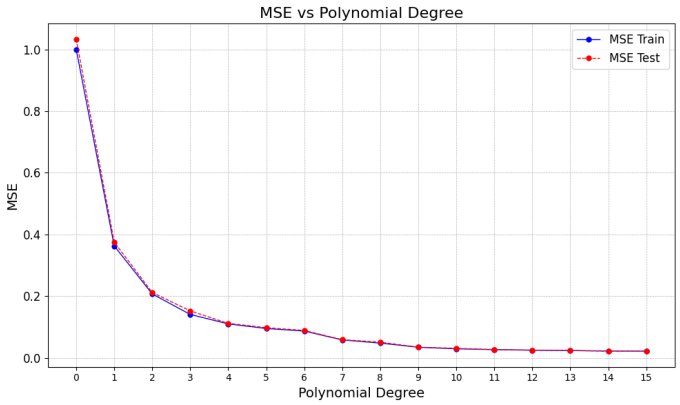


Figure 5: Line plot of MSE vs polynomial degree comparing training and test data in an OLS regression model for 10000 data points.

The MSE for both training and test data slowly decreases until stabilizing around the 9th degree.
We can see that by using a larger dataset the risk of overfitting reduces and the model is able to sustain a higher polynomial degree (from 9th) without seeing fluctuations in test error.
Still focusing on OLS regression, the next step is to assess the values of bias, variance, and error based on the complexity of our model. We choose to work with 200 bootstrap samples.

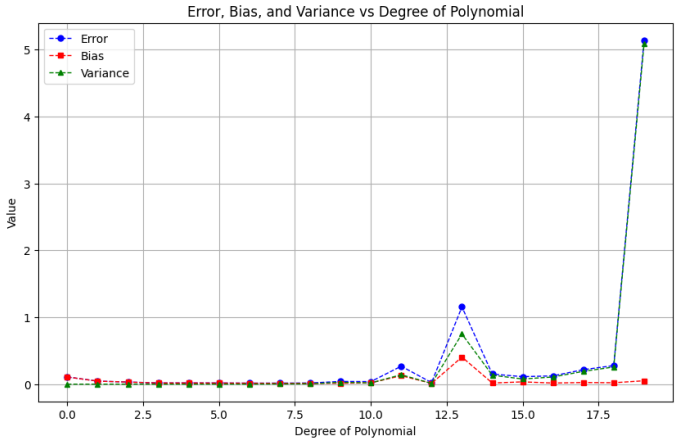
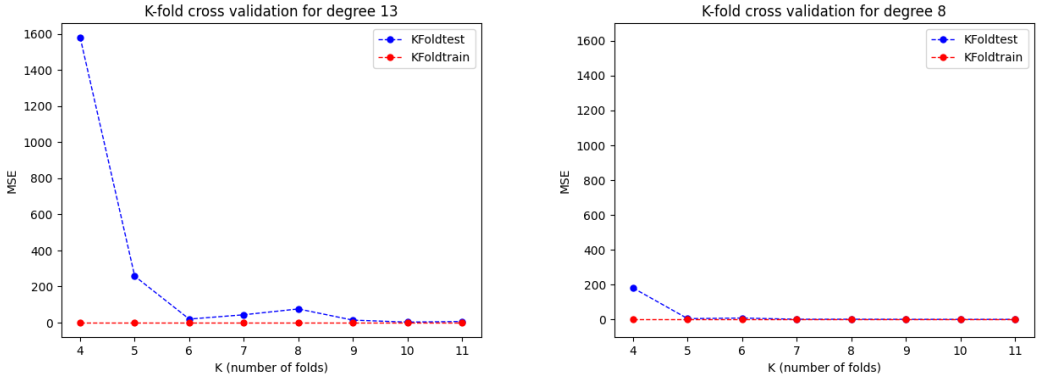


Figure 6: Line plot of Bias-Variance-MSE vs polynomial degree in an OLS regression model.

The Figure 6 shows that bias remains relatively low, with a slight spike for the 13 degree, this implies that the underfitting problem is minimal.
Initially, the error and variance are relatively low, thus around a polynomial degree of around 13 both have a peak and then fall back. Due to high variance the error increase quickly after 17th degree, which highlights a possible overfitting of the model. However from the 3rd degree till the 10th degree the model performance seems to improve slightly. The best model strikes a balance between these factors in order to reduce MSE while still generalizing well to new data.

Afterwards, we decided to continue the analysis by performing a K-fold cross validation on our data from the OLS regression.



(a) (b)

Figure 7: Line plot of MSE vs k - fold comparing training and test data in an OLS regression model for degree 13 (a) and 8(b).

Figure 7 shows the MSE value for a 8th and 13th-degree polynomial as the number of K folds varies between [4,11]. Since we had observed an increase in error from the analysis utilizing bootstrap, we made the decision to investigate the 13th degree of the polynomial in more detail. By doing this we can compare its MSE values with the ones of a model that we saw from the bias-variance trade-off had a better fitting
The MSE of the training data is low across all folds. The MSE of the test data dramatically decreased from fold 4 to fold 6, then a little increased between folds 6 and 8, then began to slow down and stabilize between folds 9 and 11.
The model's generalization improves as more training data is used, indicating a balance between complexity and training data. A slight increase between 6 and 8 may indicate a dip in general-

ization. The model's stabilization between 9 and 11 suggests optimal generalization, balancing fit and complexity with minimal performance variation across additional folds.

4.1.2 Ridge regression

To continue our analysis, we use the Ridge regression for a dataset of 400 points investigating the impact of varying the polynomial degree $\in [0, 19]$ and the regularization parameter $\lambda \in [1 \times 10^{-5}, 10]$. Heatmaps' color intensity reflects MSE, with darker colors indicating lower value and lighter ones indicating higher ones.
As for the OLS method, we implemented our own code to calculate the values of β

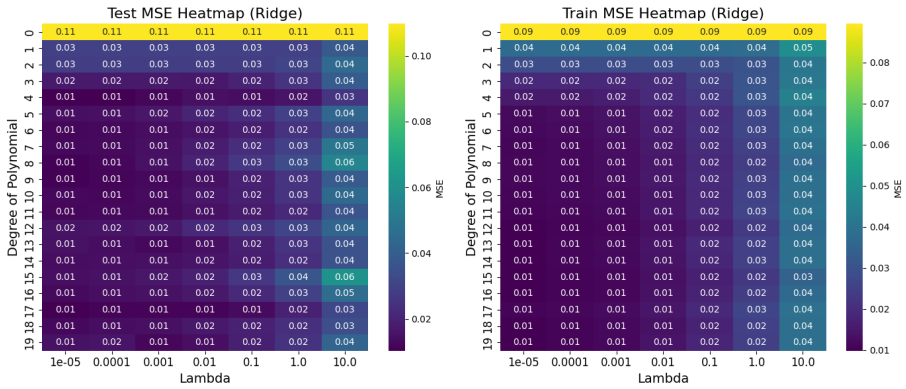


Figure 8: Heatmap of MSE for test and training data showing variation across polynomial degrees and λ values in a Ridge regression.

The test MSE heatmap, represented on the left image of Figure 8, evaluates a model's performance on unseen data. It shows that for higher polynomial degrees (around 15-19), the model achieves a low MSE, especially when λ is small (10^{-5} to 0.1). On the other hand, increasing the value of λ and decreasing the degree the MSE tends to fall slightly. This may suggests that more complex models perform well but benefit from some level of regularization to avoid overfitting.
As for the previous graph, the heatmap on the right for the training data shows that MSE slightly decrease as the polynomial degree increases and λ decreases.
These heatmaps essentially demonstrate how, particularly when dealing with complex data, striking the right balance between model complexity and regularization is crucial to gaining optimal performance.
We also decided to perform a K-fold cross validation on our data from the Ridge regression. Figure 9 shows the MSE value for a 13th-degree polynomial as the parameter lambda varies between $[1 \times 10^{-5}, 10]$ and the number of K folds varies between [4,11].

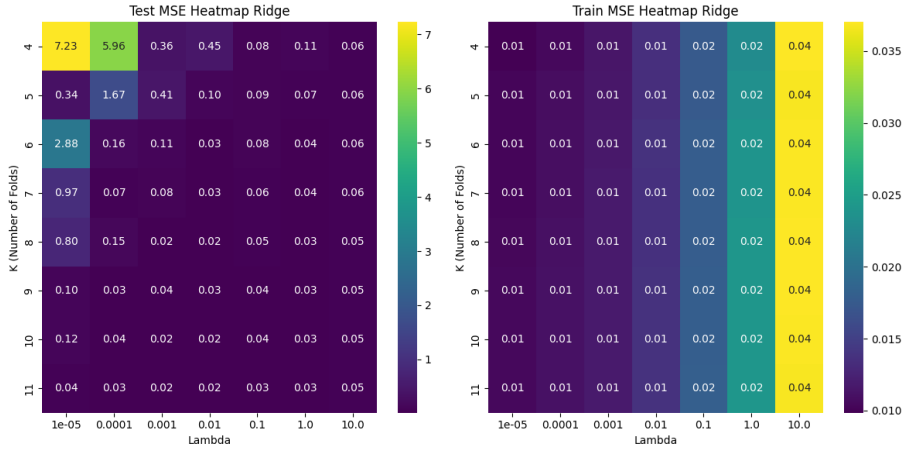


Figure 9: Heatmap of MSE for test (left) and training (right) data showing variation across number of k-fold and λ values in a Ridge regression.

The train data's MSE heatmap on the right image 9 illustrates that the model obtains a low MSE for all k-fold numbers, but only for tiny λ values ≤ 0.1. This suggests that the model is adequately fitting the training set when regularization is low, allowing it to identify complex patterns without dramatically increasing bias. However, we observe a increase in MSE as the λ value rises, indicating that excessive regularization is causing underfitting. In summary, the model performs well at low λ and becomes less stable as regularization becomes stricter.
On the other hand, as is typical in regression analysis, the test data's MSE is comparatively higher than the training data's. The model performs better on the training set than it does on unseen data, which highlights the difficulty of generalization. Additionally, when λ values drop and the number of K folds rises, the test mean square error (MSE) falls. This shows that by minimizing regularization, the model's ability to adapt to the test data's underlying structure is improved, leading to an increase in prediction accuracy.

4.1.3 Lasso regression

The same analysis can be done for Lasso regression. Since it is a bit difficult to implement our own code for the Lasso Regression, we decided to use the one provided by skit-learn. Unfortunately, due to the fact that it was taking to much time to run the code, we had to decrease the number of iterations. This caused the method to not converge and, therefore, to give imprecise results.

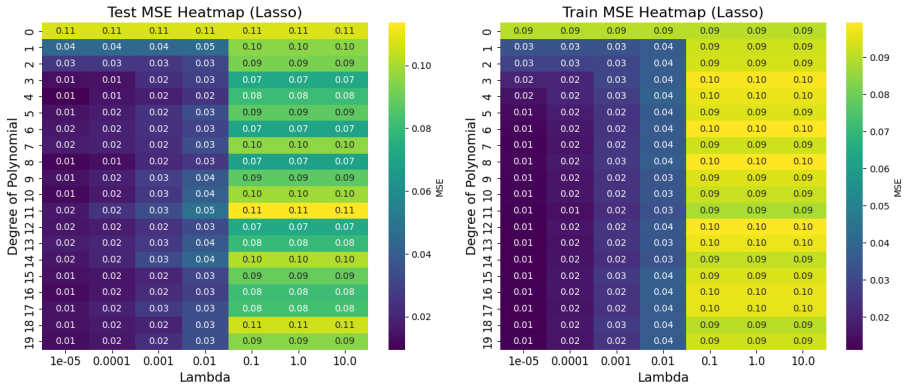


Figure 10: Heatmap of MSE for test (left) and training (right) data showing variation across polynomial degrees and λ values in a Lasso regression.

In this case, for both test and train MSE , as the degree of the polynomial increases, the model

tends to fit the data better.
For both test and train, there is a discernible difference at $\lambda = 10^{-2}$; at small lambda values, the MSE is small. As lambda increases, regularization becomes stronger, causing underfitting, as seen by raising the MSE value.
In essence, smaller lambda values and higher polynomial degrees yield the best results, suggesting that the regularization strength and model complexity must be balanced.
We also performed a K-fold cross validation for the data from the Lasso regression. Figure 11 shows the MSE value for a 13th-degree polynomial as the parameter lambda varies between [1×10^{-5}, 10] and the number of K folds varies between [4,11].

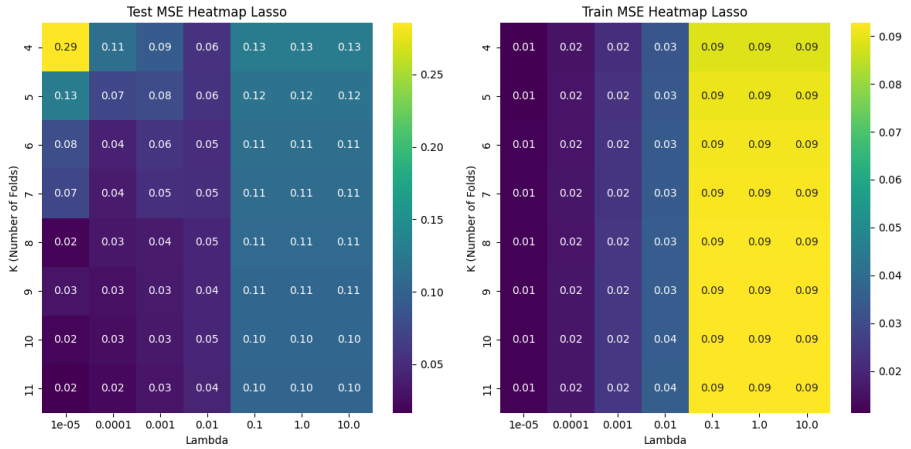


Figure 11: Heatmap of MSE for test (left) and training (right) data showing variation across polynomial degrees and λ values in a Lasso regression cross-validating data with k- fold.

The train data's MSE heatmap on the right image 11 shows that the MSE varies based on the value of lambda rather than changing as the number of k-fold changes. The MSE assumes a value of 0.09 for lambda values greater than 10^{-2}. A stronger assessment of the test MSE is produced by increasing the number of k-folds, which aids in obtaining more accurate estimations of model performance.
To summarize, the findings highlight that although regularization is essential for managing model complexity and mitigating overfitting, employing an increased number of k-folds can also improve model assessment and generalization, especially at lower λ values when the model is less restricted.

4.2 Terrain data

4.2.1 OLS

We developed our own code to implement a function for the OLS model for the terrain data.
We chose to scale our data because it assures that all variables are on the same scale, reduces the potential for bias resulting from possible outlier, and facilitates data interpretation.
Fig. 12 shows the MSE values for the training and testing data of the Terrain data across polynomial degrees from 0 to 15.

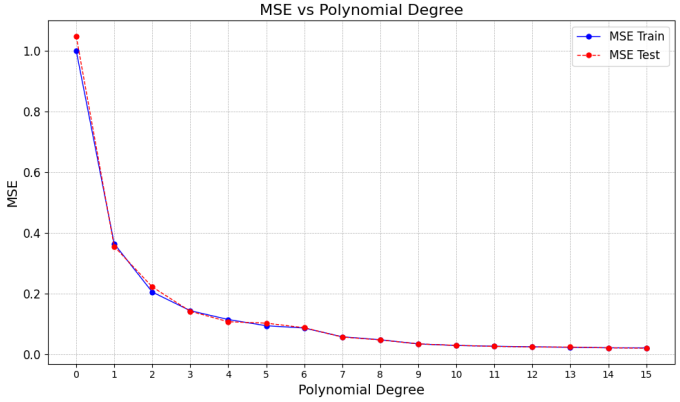


Figure 12: Line plot of MSE vs polynomial degree comparing training and test data in an OLS regression model with 10000 data points.

For both training and test data, the MSE gradually drops until it stabilizes at the ninth degree, demonstrating how the model with higher degree polynomials seems to suit the data better.

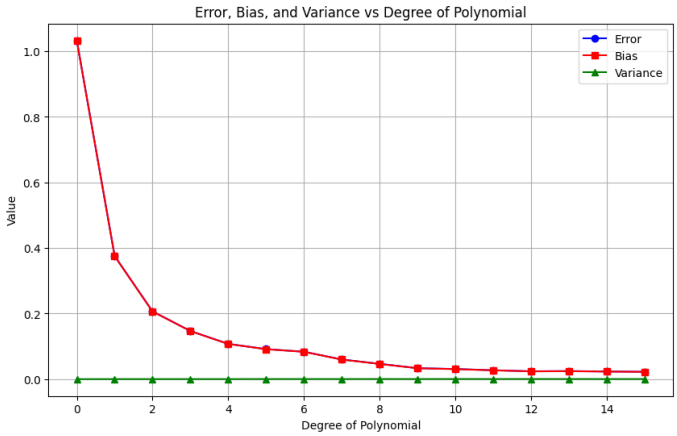


Figure 13: Line plot of Bias-Variance-MSE vs polynomial degree for terrain data in an OLS regression model.

The Bias-Variance-MSE plot (Fig. 13) show that the error decreases as the degree of the polynomial increases, stabilizing at a low value beyond degree 9. The bias follows a similar decreasing trend, also reaching near-zero values for higher degrees. The variance remains consistently low across all degrees with minimal change, suggesting that increasing model complexity does not lead to significant overfitting.
Figure 14 shows the MSE value for a 9th-degree polynomial as the number of K-folds varies between 4 and 11. The 9th-polynomial was chosen because this is the polynomial were the MSE stabilized at a low value.

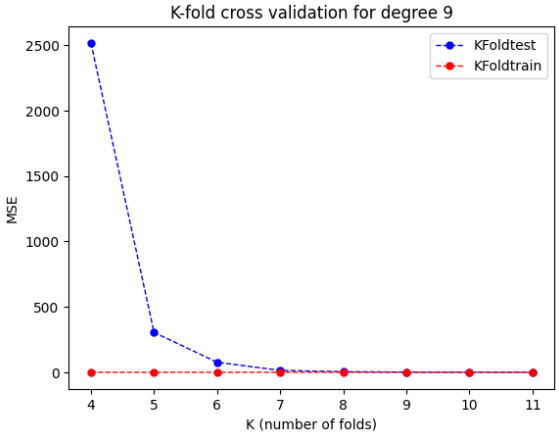


Figure 14: Line plot of MSE vs k - fold comparing training and test data in the terrain OLS model.

The results show that the MSE for the test data decreases significantly as the number of folds increases from 4 to 6, after which it stabilizes at a low value. The MSE for the training data remains consistently low across all values of K, showing minimal variation.

4.2.2 Ridge regression

Using the Ridge regression, we further analyze the effects of changing the regularization parameter $\lambda \in [1 \times 10^{-5}, 10]$ and the polynomial degree $\in [0, 15]$.

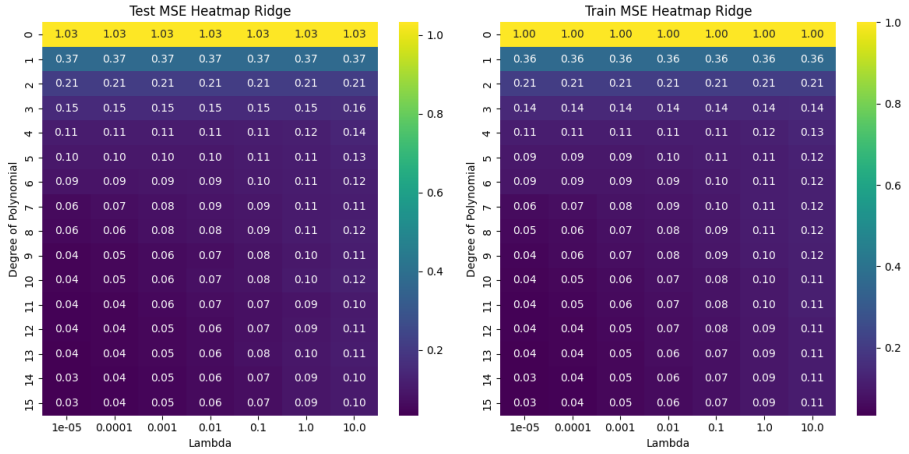


Figure 15: Heatmap of MSE for test and training data showing variation across polynomial degrees and λ values in a Ridge regression.

The MSE lowers as the polynomial degree increases and λ falls, according to both test and train heatmaps, represented on the Figure 15. The polynomial degree in particular has a significant influence; in fact, the MSE is greater than 0.2 for polynomials smaller than the 3th degree. Compared to the training data, the test data's MSE value is relatively bigger.

We also created a heatmap showing how the MSE varying lambdas and varying number of k folds for a 9th degree polynomial (Fig. 16).

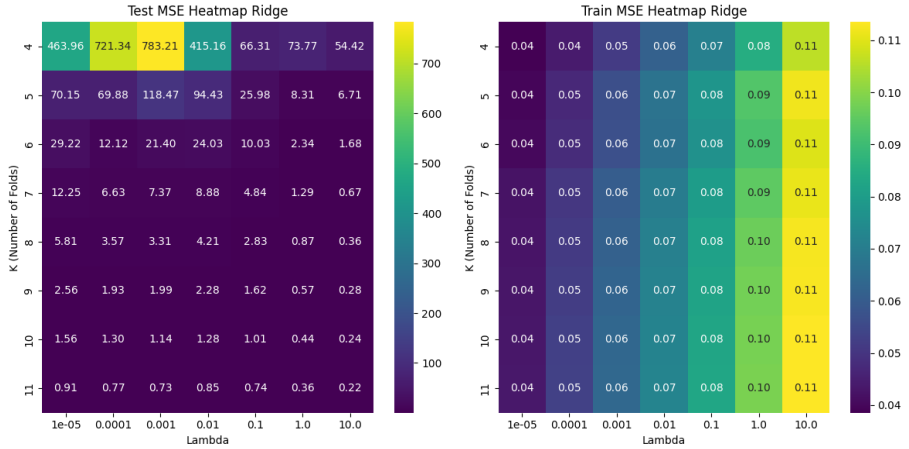


Figure 16: Heatmap of MSE for test (left) and training (right) data showing variation across number of k-fold and λ values in a Ridge regression.

The training MSE (right) the MSE is mostly dependent on the value of λ, with the lowest MSE values at $\lambda = 10^{-5}$, suggesting that underfitting is being caused by excessive regularization (elevate value of lambda). The training MSE also seems to decrease at the lowest number of folds (K=4).
The test MSE (left) decreases significantly with an increase in both the number of folds (K), suggesting that raising the k-fold helps in producing estimates of model performance that are more accurate.

4.2.3 Lasso regression

We did the same analysis for Lasso regression.

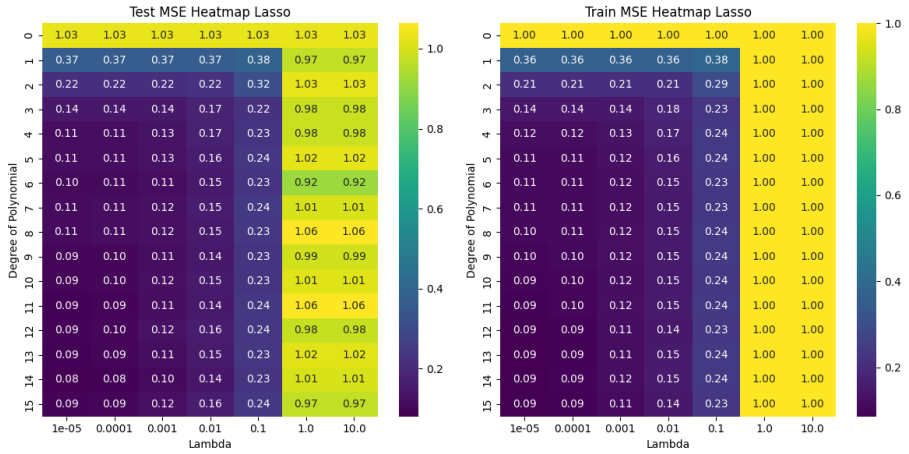


Figure 17: Heatmap of MSE for test (left) and training (right) data showing variation across polynomial degrees and λ values in a Lasso regression.

Lambda appears to have a significant effect on the MSE for both the train and test heatmaps, represented on Figure 17. When $\lambda > 0.1$, the MSE rises and approaches 1.

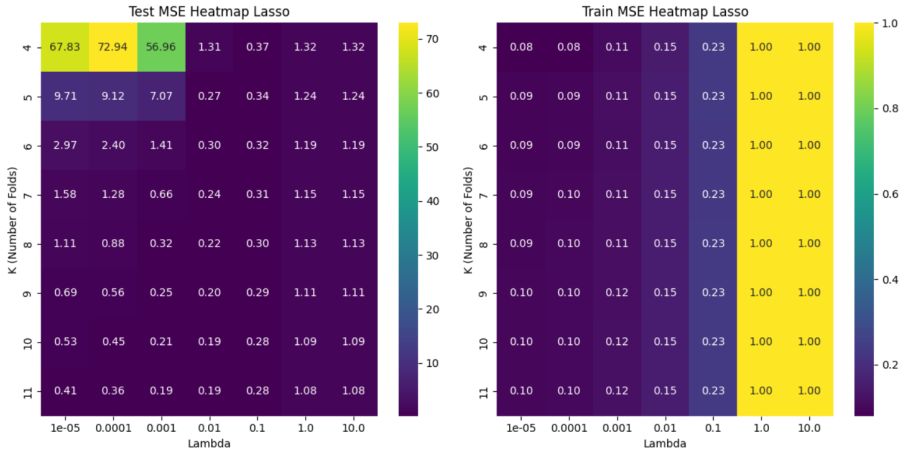


Figure 18: Heatmap of MSE for test (left) and training (right) data showing variation across polynomial degrees and λ values in a Lasso regression cross-validating data with k- fold.

The Lasso MSE - K-fold - Lambda heatmaps made for a 9th degree polynomnial (Fig. 18) shows that the test MSE (left) decreases with an increase in both the number of folds and the lambda value, reaching the lowest MSE values for K=11 and lower lambda values (0.01 and below). The train MSE (right) shows the same trend as the Ridge MSE heatmap (Fig. 16), where the MSE is mostly dependent on the value of λ, but with a slight decrease in MSE at the lowest K-fold number (K=4)

5 Conclusion

The goal of our statistical analysis was to determine the best model by evaluating the outcomes of two types of data using three distinct linear regression techniques (Ordinary Least Squared (OLS), Ridge, and Lasso). To improve the accuracy of the results we obtained, we additionally investigated and implemented resampling techniques, including k-fold cross validation and bootstrapping.
In our analysis of the Franke function, the results suggest that Ridge regression fits the underlying data better than the other two methods, since it reduces the mean squared error.
Though Lasso's findings seem to be superior than Ridge's in the K-fold cross-validation results, we prefer to rely on Ridge because we ran less iterations for Lasso. Due to the possibility that this limited Lasso's performance, Ridge is the more reliable alternative in this instance. However, with more iterations, the Lasso model's performance might get better, allowing for greater result optimization.
Comparing the results obtained with 400 or 10,000 data points for OLS, we can notice how by increasing the number of datas we can schive a lower discrepancy between the MSE values of the test and train data. Specifically, the MSE values for the test data seem to stabilize and oscillate less. Therefore we could have probably achieved more accurate results by increasing the number of data points. However, this would have been too computationally expensive. About our analysis of Stavanger, Norway's landscape, which is comprised of a more intricate dataset, we found that the best fit in this instance was the Ordinary Least Squares fit, which produced generally lower MSE values in relation to Ridge and Lasso.
We observed that for the OLS, model quality rises as model complexity does. We think that a higher level of complexity may have produced the best results. However, due to computational performance problems we did not investigate that further
When comparing the OLS model for both the Franke function data and the terrain data using the bootstrap resampling method with the K-fold method, we found that the model performs better using the bootstrap method. In comparison to K-fold, bootstrap consistently yielded lower MSE values after 200 iterations for both datasets, suggesting a better model fit.

References

[1] A. C. Müller and S. Guido, "Linear models: From simple to multiple regression," in *Introduction to Machine Learning with Python: A Guide for Data Scientists*, pp. 33–47, Cham: Springer Nature Switzerland, 2023.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer, second ed., 2009.

[3] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, Massachusetts: MIT Press, 2012.

[4] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx, *Regression: Models, Methods and Applications*. Springer-Verlag Berlin Heidelberg, 2013.

[5] ChatGPT, "Chatgpt conversation on train and test split," 2024. Accessed: 2024-10-03.

[6] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[8] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

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