Regression and Resampling Methods: Topographical Analysis with Ordinary Least Squares, Ridge and Lasso Regression

Jonas Jørgensen Telle[1], Marius Torsheim[1], and Maria Klüwer Øvrebø[2]
[1]*Department of Physics, University of Oslo, N-0316 Oslo, Norway*
[2]*Department of Mathematics, University of Oslo, N-0316 Oslo, Norway*
(Dated: October 4, 2024)

Abstract: We explore the trade-off between computational cost and goodness-of-fit for different polynomial regression models. We first fit synthetic data (Franke's function) and proceed to real-word topographical data from Glittertind in Lom, Norway. We use Ordinary Least Squares (OLS), Ridge and Lasso regression, testing selected hyperparameters $\lambda \in [10^{-4},\ 1]$ with k-fold cross-validation. For both datasets, we find that smaller penalties are better, with OLS giving the best fit and Lasso performing significantly worse. For degrees $p \leq 5$ with k-fold optimized parameters, we use bootstrap to estimate the bias and variance, finding that higher complexities drastically lower the bias while only slightly increasing the variance. Finally, we use a polynomial degree 30 to fit the topographical data with OLS and Ridge and discuss briefly how such findings can lead to better terrain models and improved predictive accuracy for geological models.

I. Introduction

Regression analysis is a fundamental tool in statistical modeling, widely applied in both scientific research and related industry. Ordinary Least Squares (OLS) is one of the simplest and most well-known regression techniques, while more advanced methods like Ridge and Lasso introduce regularization, addressing overfitting by incorporating bias (penalty) terms. In theory, these methods might result in better performance and generalizability, especially when dealing with multicollinearity or high-dimensional data.

In this paper, we explore various regression techniques in the context of machine learning, focusing on OLS, Ridge, and Lasso regression. We begin with the implementation of regression models to fit the Franke function, a widely used benchmark function in numerical analysis. This provides useful intuition and insights for the following analysis on real data. The goal is to determine which models perform best, taking both MSE and runtime into consideration.

In addition to understanding regression, we delve into the bias-variance tradeoff, a key concept in machine learning that highlights the balance between model complexity and prediction accuracy. To analyze this tradeoff, we apply resampling techniques such as bootstrapping and cross-validation, which help assess the reliability and stability of the models.

Finally we apply these regression models to real-world topographical data of the Norwegian mountain Glittertind. This dramatic landscape provides challenging testing ground for our regression algorithms due to its complexity. Our goal is to find a good trade-off between model complexity, which leads to higher computation costs, and goodness-of-fit. The ability to efficiently fit terrain data could accelerate geological studies, where one might try to find general trends in a landscape. Such information could both give insight into how the region developed and looked historically, and help in mapping deep sea surfaces, limiting the number of needed datapoints where measurements are expensive.

In Section II, we derive expressions for expectation and variance of OLS, in addition to the bias-variance decomposition. Section III presents the methodology used for implementing the regression models in depth, along with a further discussion of the bias-variance tradeoff and the use of resampling techniques. In Section IV, we analyze the results obtained from applying these methods to both Franke's function and real-world terrain data. Section V provides a discussion of these results, followed by conclusions and ideas for future work in Section VI.

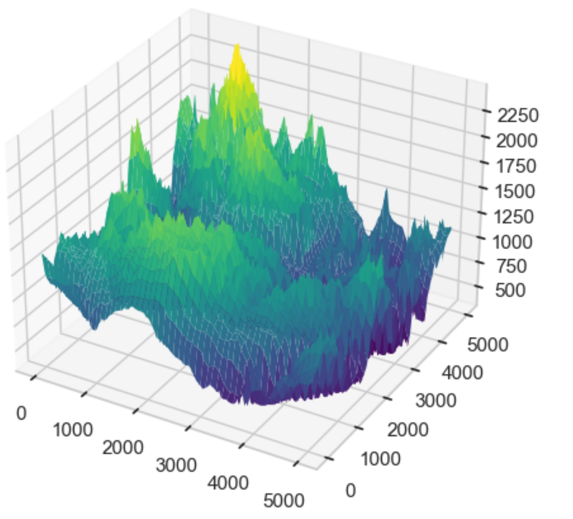

Figure 1. 3D plot of the extended dataset of the area around the mountain Glittertind in Lom, Norway, the yellow peak visible in the plot.

II. Theory

In the analysis of our data, we assume that there exists a continuous underlying function $f(\boldsymbol{x})$ and an error term, $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$, that describes our data such that

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

We approximate this function with our model $\tilde{\boldsymbol{y}}$. The expectation value of $\boldsymbol{y}$ for each index i is then given by

$$\begin{aligned} E(y_i) &= E[f(x_i) + \epsilon] \\ &= E[\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{n-1} x_{in-1} + \epsilon] \\ &= E\left[\sum_{j=0}^{n-1} \beta_j x_{ij}\right] + E(\epsilon) \\ &= \sum_{j=0}^{n-1} \beta_j x_{ij} \end{aligned}$$

where we use the linearity of expectation. The variance of $\boldsymbol{y}$ for a index i is

$$\begin{aligned} V(y_i) &= V[f(x_i) + \epsilon] \\ &= V[\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{n-1} x_{in-1} + \epsilon] \\ &= V[\beta_0 + \beta_1 x_{i1} + \cdots + \beta_{n-1} x_{in-1}] + V(\epsilon) \\ &= V(\epsilon) \\ &= \sigma^2 \end{aligned}$$

where we use that the variance is invariant.

From these results it follows that $y_i \sim N(X_{i,*}\beta, \sigma^2)$. Using this, we find that the expectation value for the OLS expressions for the optimal parameter $\hat{\beta}$ is given as

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

where we again use the linearity of the expectation. This shows that $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of $\boldsymbol{\beta}$.

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

$$\begin{aligned} V(\hat{\boldsymbol{\beta}}) &= V\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\right] \\ &= \left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\right] \; V(\boldsymbol{y}) \; \left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\right]^T \\ &= \left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\right] \; \sigma^2\boldsymbol{I} \; \left[\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right] \\ &= \sigma^2\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right] \\ &= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \end{aligned}$$

where we use that if $\boldsymbol{A}$ is a constant matrix and $\boldsymbol{x}$ is a vector, then $V(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}V(\boldsymbol{x})\boldsymbol{A}^T$.

We see that the variance is dependent on the inherent error σ^2 and $\boldsymbol{X}^T\boldsymbol{X}$. Thus, if there is less noise in the data we get more precise parameter estimates. Additionally, the values in $\boldsymbol{X}^T\boldsymbol{X}$ reflects both the sample size and the spread of the predictors, which also affect the variance.

The mean squared error can be decomposed into a three terms, the bias term which measures the mean deviation from the true data, the variance term that contains the variance of the model itself and a term for the variance of the noise. The bias-variance decomposition can be derived as follows:

$$\begin{aligned} E\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] &= E\left[(\boldsymbol{y} - E(\tilde{\boldsymbol{y}}) + E(\tilde{\boldsymbol{y}}) - \tilde{\boldsymbol{y}})^2\right] \\ &= E\left[(\boldsymbol{y} - E(\tilde{\boldsymbol{y}}))^2 + 2(\boldsymbol{y} - E(\tilde{\boldsymbol{y}}))(E(\tilde{\boldsymbol{y}}) - \tilde{\boldsymbol{y}}) + (\tilde{\boldsymbol{y}} - E(\tilde{\boldsymbol{y}}))^2\right] \\ &= E\left[(\boldsymbol{y} - E(\tilde{\boldsymbol{y}}))^2\right] + 2E\left[(\boldsymbol{y} - E(\tilde{\boldsymbol{y}}))(E(\tilde{\boldsymbol{y}}) - \tilde{\boldsymbol{y}})\right] + E\left[(\tilde{\boldsymbol{y}} - E(\tilde{\boldsymbol{y}}))^2\right] \qquad (1) \end{aligned}$$

The first term in the expression above can be simplified as follows:

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

Where we've used that $E[E[\boldsymbol{x}]] = E[\boldsymbol{x}]$ and that $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$. Substituting this simplification into (1), we get:

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

where we use the standard definition of $Bias[\tilde{\boldsymbol{y}}] = f(\boldsymbol{x}) - E(\tilde{\boldsymbol{y}})$.

III. Method

A. Regression on the Franke Function

In this section, we perform regression analysis on the Franke function using three different techniques: Ordinary Least Squares (OLS), Ridge regression, and Lasso regression. Each method introduces its own approach for estimating the regression coefficients, with Ridge and Lasso incorporating regularization to address potential issues of overfitting and multicollinearity. The Franke function provides synthetic data to test our models on a surface fitting task, and MSE and R^2 scores are used as performance metrics.

For each model, we created our artificial data with the Franke function $f(x, y)$, where the two input variables x and y were uniformly spaced in the interval [0,1]. To simulate real-world noise, we added a standard Gaussian term ($\epsilon \sim N(0, 1)$), with an adjustable noise coefficient ($k = 0.1$). We then created our polynomial design matrix by expanding the input features x and y up to the fifth degree, and used the `train_test_split` function from `scikit-learn` to divide the dataset into training (4/5) and testing (1/5) subsets [1].

Finally, we perform scaling of the data. In theory, scaling may improve stability with polynomial fits, since the features have different orders so that large numbers blow up for higher order fits. Especially for Lasso, if some features get very large, they will likely be set to zero. With this in mind, we center the data on OLS and Ridge, while we use `StandardScaler` from `scikit-learn`, which also standardizes the value range, on Lasso [1]. However, we do not find that scaling has a significant effect on the results, which makes sense seeing as the range of values from the Franke function is limited when the inputs are limited to the interval [0,1].

1. *Ordinary Least Squares*

Using OLS, we estimated the coefficients β by solving the standard system of equations:

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

where $\tilde{\boldsymbol{y}}$ denotes the predicted values, $\boldsymbol{X}$ is the design matrix containing the polynomial features, and $\boldsymbol{\beta}$ is the vector of regression coefficients. We solve these equations analytically with matrices in NumPy [2].

The model was trained for polynomial degrees ranging from 0 to 5, and the values of the regression coefficients $\boldsymbol{\beta}$ were recorded (figure 7). We implemented a function to create the two-dimensional design matrix, and we exclude the intercept in training. The trivial case with degree 0 simply returns the mean of the training data. The MSE and R^2 scores were computed for both the training and test sets, and the results were visualized to assess the relationship between model complexity and performance.

Model performance was evaluated using the following metrics:

- **Mean Squared Error (MSE)**: This is also our OLS cost function, and measures the average squared difference between the observed values y_i and predicted values $\tilde{y}_i$, given by:

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

- R^2 **Score**: The coefficient of determination R^2, which quantifies the proportion of variance explained by the model:

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

where $\bar{y}$ is the mean of the observed values.

2. *Ridge Regression*

In the next phase, we extended the OLS method by incorporating Ridge regression, which adds a regularization term to the standard OLS cost function:

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p}\beta_j^2$$

This may mitigate overfitting, especially with high-degree polynomial features, by penalizing large regression coefficients [3]. The shrinkage of coefficients is illustrated in figure 9. From the modified cost function, we obtain an analytical expression for the Ridge regression model:

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

where λ is the regularization parameter that controls the penalty on the magnitude of the coefficients, and $\boldsymbol{I}$ is the identity matrix. Using NumPy, we solve this equation to fit our model analytically [2].

Historically, Ridge regression was a simple stability trick to avoid singular matrices, but its shrinkage properties can prove useful also when the data is not perfectly multicollinear [4]. The hope is that high-variance parameters will be penalized, reducing variance at the cost of the small bias introduced by the addition of the hyperparameter term. This bias does mean that, unlike OLS, ridge regression will not give a perfect fit even without noise when our target is actually a polynomial of same or lesser degree that our model complexity, but the bias should be small and worth it in real world applications with noise where we want to avoid overfitting.

We tested Ridge regression for various values of λ to observe its effect on model performance, evaluating both the MSE and R^2 scores across the training and test sets. The dataset was split as before, and the data was centered before fitting.

3. *Lasso Regression*

Lasso regression includes an ℓ_1-norm regularization, which encourages sparsity in the solution by shrinking some coefficients all the way to zero. The Lasso model minimizes the following cost function:

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} |\beta_j|$$

where λ controls the strength of the ℓ_1 penalty. This leads to models with fewer, but perhaps more meaningful, features due to feature selection, but is not solvable analytically [3].

We therefore used `scikit-learn`'s `Lasso` implementation, which employs coordinate descent for optimization [1]. Utilizing `scikit-learn`'s `Pipeline` functionality, we used `PolynomialFeatures` to create the design matrix and `StandardScaler` to center and normalize the features for our Lasso implementation. Note that this differs from the previous regression methods, for which we wrote our own code for the design matrices and centering.

Lasso regression completely zeros out coefficients, and while this might be useful on terrain data to find the overarching trends in the terrain, it did not improve performance on the Franke function. The effect of Lasso-regularisation is illustrated in figure 9. A thorough comparison of the methods on our datasets will be given in the Disussion.

B. Bias-Variance Trade-off and Resampling Techniques

1. *Bias-Variance Trade-off*

In the Theory section, we obtained the following decomposition of the expected error, which underlies the bias-variance tradeoff:

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

where:

- **Bias** measures the error introduced by the assumptions of the model. It represents how far the model's predictions are from the true values, on average.
- **Variance** (V) measures the model's sensitivity to the specific dataset used. It represents how much the predictions would vary if we used different training sets.
- σ^2 represents the irreducible noise, which is the variance inherent in the data which cannot be reduced by the model.

Ideally, we want a low error, and thus a low bias and variance. In practice, a low bias corresponds to a good mean fit, while a low variance corresponds to a low spread. A high spread is bad, indicating that the model is unstable and will not generalize well. This is why it is essential to split the data, else the final model score would not include the error due to the variance.

With high complexities, the bias tends to be low, since the model fits the training-data very well and we expect that the noise is symmetrically distributed. Thus, the deviations in different directions cancel out. Still, with high complexities and comparatively small datasets, the model will be very sensitive to the training data, passing through every point. This results in a high variance, and poor test-scores.

In figure 2, we illustrate how variance increases with higher complexities. In creating this figure, we used artificial data with normally distributed noise, which is symmetric so that the training error closely reflects the model bias. Indeed, the training error decreases with complexity. However, the variance increases at high complexities so that the test error becomes large. To get consistent results and a smoother curve, we take the average over multiple datasets, so that each point on the graph corresponds to an average performance of a polynomial OLS regression of the given degree.

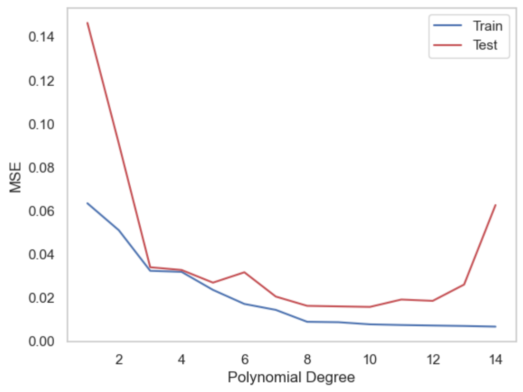

Figure 2. *Reproduction of figure 2.11 from Hastie, Tibshirani, and Friedman [3], illustrating the training and test errors as a function of polynomial degree. The test error initially decreases as model complexity increases, but rises again due to overfitting at higher degrees. The training error continuously decreases, reflecting improved fit to the training data.*

2. *Bootstrapping*

We obtained an estimate of the variance by fitting models on different sub-datasets with the same method. If the predictions agree closely, the variance is low. We did this with bootstrap, using the following steps:

1. Randomly generated B bootstrap samples, drawing with replacement to resample the original training

data.

2. For each bootstrap sample, fitted the OLS model on the dataset and made a prediction for each data-point in the test data, getting n predictions.

3. For each data point, calculated the squared bias as the squared difference between the mean of the predictions from all B models and the true value. The final squared bias estimate is the mean of this distribution of n bias estimates.

4. For each data point in the validation data, calculated the variance in the predictions from all B models. The final variance estimate is the mean of this distribution of n variance estimates.

Our bootstrap implementation is based on code provided in [4].

3. *k-Fold Cross-Validation and Grid Search*

We implemented the k-fold cross-validation method as a resampling technique to evaluate the performance of different regression models. Cross-validation is a widely used method for estimating the generalization error of a model by partitioning the dataset into k subsets (folds) and iteratively training the model on $k-1$ folds while testing on the remaining fold [3]. This process is repeated k times, with each fold being used as the test set once, and the mean squared error (MSE) is averaged over all iterations.

We implemented this method for all regression methods using the `cross_val_score` functionality provided by the `scikit-learn` library [1]. After implementing cross-validation, we compared the MSE obtained using this method with the MSE obtained from the bootstrap resampling technique with OLS.

For Ridge and Lasso, we performed 10-fold cross-validation and computed the average MSE for different values of the regularization parameters. The cross-validation method helped us select optimal regularization parameters by finding the value of λ that minimizes the MSE on the validation data and using only this model on the test data. In testing using fewer folds, we found no significant difference on the Franke function, and so decided to use 10-folds for the topographical data. In addition to `cross_val_score` we also implemented a grid search using `GridSearchCV` from the `scikit-learn` library [1]. This method calculates the cross validation score for a grid of values and returns the optimal parameters, and proved especially useful on the topographical data.

C. Fitting topographical data

1. *Downloading and formatting data*

For the analysis of topographical data, we used digital terrain data from the website Geonorge. This data gives the elevation of a specific area in Norway for each 10m by 10m square, and we used the DTM 10, UTM33 terrain model. The files were downloaded in a `TIFF` format.

For this project we chose terrain data for the Norwegian mountain Glittertind. The terrain data in `GeoTIFF` format were loaded using the `imageio.v2.imread` function. The data were then exported to a npy-file for further use.

2. *Topographical analysis*

Once the data had been properly downloaded and were in the correct format, we applied all three methods to this dataset. Cross-validation were used to evaluate the performance of each model and to identify which model performed the best for each degree up to degree five. A bootstrap estimate of the bias, variance and error were also obtained for the best model of each degree. The implementation of OLS, Ridge and Lasso used `Pipeline` and the `make_pipeline`-functionality, and the scaling used were `StandardScaler`, all of which are part of the `scikit-learn` library [1]. Gridsearch were then used to find the best parameters for each regression method, as well as the best method overall. This were implemented with inspiration from code found in *Machine Learning with PyTorch and Scikit-Learn* by Raschka et al [5].

While our general discussion on scaling on the Franke function also applied here, we might expected that the dramatic range in height in the topographical data would increase the importance of scaling. This is why we also standardized the values in the OLS and Ridge fits, rather than only centering as we did on the Franke function. Figure 1 indicated that, while the landscape is dramatic, there were no clear outliers in terms of single, isolated data points which would lead to problems with the scaling. Very basic tests without scaling indicated that scaling did not have a significant impact on the final model scores, though Lasso seemed more sensitive than the other methods.

Finally, we made a high-degree fit up to degree 30 for OLS and Ridge. Lasso was not included in this fit since it showed little promise in terms of goodness-of-fit on lower degrees, as well as proving computationally intensive. The fact that Ridge, with milder regularization than Lasso, increasingly under-performs at higher degrees as compared to OLS (figure 13) gives us confidence that excluding Lasso was well justified. Gridsearch were again used to find the best regression method and the model parameters of degree and, for Ridge, regularization.

IV. Results

A. Regression on the Franke Function

1. Ordinary Least Squares

Polynomial regression were performed using OLS for polynomial degrees ranging from one to 15. Figure 3 displays the training and test MSE as a function of polynomial degree. Both errors decrease significantly up to around degree five, where the MSE stabilizes near 0.01, which is the square of the noise coefficient k. We see minimal improvement beyond this point. The errors for both the training and test datasets are closely aligned for all degrees, perhaps barely starting to come apart at degree 15, suggesting that the model generalizes well to the test data without signs of significant overfitting.

Figure 4 expands on this, showing the bias, variance, and estimated error using bootstrap resampling for OLS on the Franke function. As the polynomial degree increases, the bias decreases, but we now see a clear trend in the variance. However, note the order of magnitude of 10^{-5}; even though this figure shows that the variance increases with model complexity as we would expect, the variance is so low that it makes no difference to the overall fit.

Figure 5 illustrates the behavior of the training and test MSE as the number of data-points increases. Note that since the Franke function is a function of both x and y, a meshgrid was created, so that 100 datapoints in the plot corresponds to $100^2 = 10,000$ datapoints in 2D. With few data-points, the test MSE is relatively high. As the number of data-points increases, the test MSE drops sharply until around 100 data points, where training and test MSE stabilize around 0.012. The importance of sufficient data-density is an important lesson for the fitting of real data.

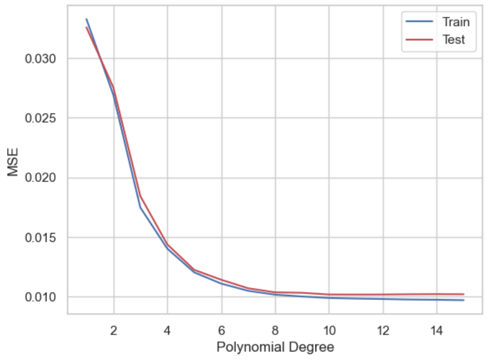

Figure 3. Training and test errors as a function of polynomial degree for the Franke function. MSE decreases sharply up to degree five, then stabilizes around 0.01, indicating no significant accuracy gains beyond this point. The close alignment of training and test errors suggests minimal overfitting.

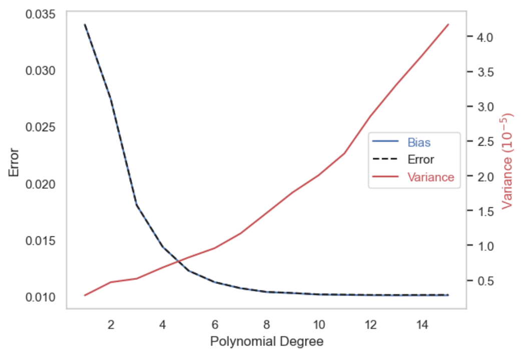

Figure 4. Mean squared error, bias, and variance as a function of polynomial degree for the Franke function, using bootstrap resampling. The plot shows that as model complexity increases, bias decreases significantly, while variance increases. The MSE stabilizes after a certain degree, indicating diminishing returns from further increases in model complexity.

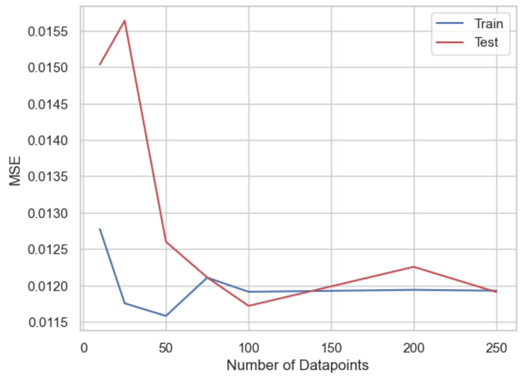

Figure 5. MSE as a function of the number of data points for the Franke function. The MSE fluctuates for both training and test datasets with fewer than 100 data points, indicating instability. Beyond 100 data points, MSE stabilizes, with training and test errors closely aligned, suggesting better generalization.

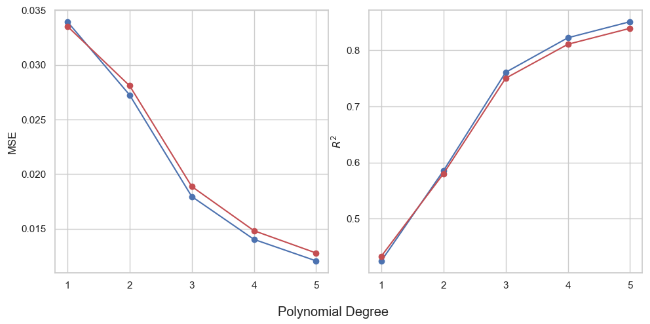

Figure 6. On the left, MSE is plotted against polynomial degree for the Franke function using OLS. The close alignment of training and test errors suggests good generalization without overfitting. On the right, we show the R^2-score for the same models.

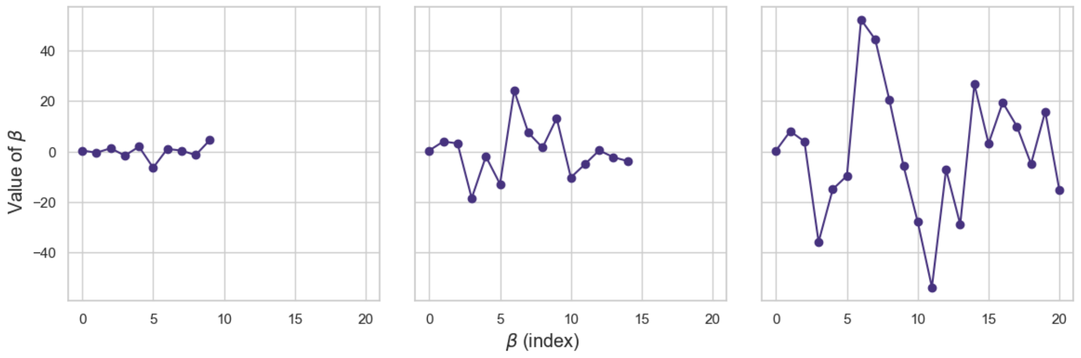

Figure 7. Variations in the β coefficients for OLS with polynomial degrees three, four and five. We see that the coefficients vary quite a lot between the different models, and that some coefficients vary more than others.

The effect of polynomial degree on both MSE and R^2 for OLS is displayed in figure 6. Both plots show that for such low complexities, the accuracy gain of increasing model complexity is well worth it. Note that Franke's function seems to be relatively simple, so that a degree five polynomial fit gives a low MSE and an R^2-score close to 1, suggesting that the model already explains most of the variance in the data. This once again explains the plateau observed in figure 4, since the error is mostly due to the inherent variance in the data (σ^2) after degree five.

Figure 7 shows the beta coefficients of OLS for polynomial degrees three, four, and five. We can see that there is considerable variance in the coefficients between degrees, especially apparent when comparing degree four and five. Interestingly, the coefficients seem to spread out as the degree increases, in the sense that deviations from zero increase. Still, the coefficients are balancing each other out to retain a mean value close to zero. Note also that with multiple dimensions, increasing the polynomial degree by one increases the number of features dramatically. To be precise, a polynomial matrix of degree d has $p = \frac{1}{2}(d+1)(d+2)$ features.

2. *Ridge and Lasso regression*

For Ridge and Lasso regression, we found MSE for a grid of polynomial degrees and regularization parameters λ, as displayed in figure 8. Note that the error is scaled by 10^3, so that the best MSE is about 0.0128 for Ridge and 0.0149 for Lasso, with both performing best with the smallest lambda value of $\lambda = 10^{-4}$.

In figures 7 and 9, we show the coefficients for models of degree three, four and five, making the shrinkage properties of Ridge and Lasso clear. Note the similarity between low-regularization Ridge coefficients and the OLS-coefficients of figure 7, and the dissimilarity with the Lasso-coefficients. As λ increases, Ridge-coefficients are clearly suppressed, but not all the way to zero, such as is the case for Lasso. Note further the different scaling on the y-axis for Lasso, having coefficients max at around ± 3 instead of ± 40 as is the case with Ridge and OLS. This dramatic shrinkage is due to the ℓ_1-norm regularization, and does not seem to be due to the different scaling-approaches, remaining similar without scaling. To further solidify this, observe that while the magnitude of OLS and Ridge coefficients increase dramatically from degree four to five, Lasso coefficients decrease, and that Lasso-coefficients remain more similar across degrees. This consistency across degrees might support our notion that Lasso picks out the most important features.

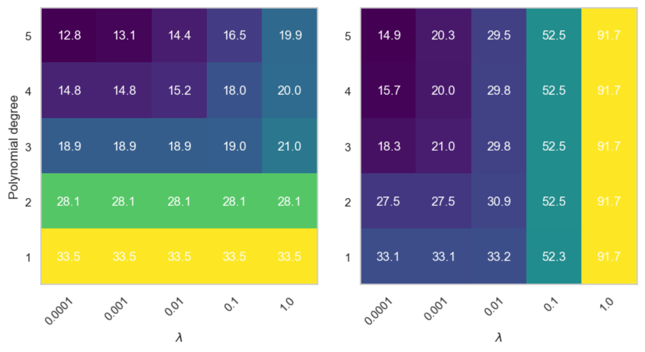

Figure 8. MSE, multiplied by 10^3, as a function of regularization parameter λ and polynomial degree for the Franke function using Ridge regression on the left and Lasso on the right. Notice that the primary predictor of performance for Ridge is the polynomial degree, with a low degree causing poor performance no matter the λ-value, while the regularization parameter is the primary predictor for Lasso, so that a high regularization flattens the model no matter the degree. Notice also that the effect of the regularization increases with model complexity. Do note that the color-map is relative to each plot, and not common to them both.

3. *Resampling Techniques*

We used bootstrapping and cross validation as resampling techniques. Bootstrapping was only used for OLS, with scores provided in figure 4. For polynomial

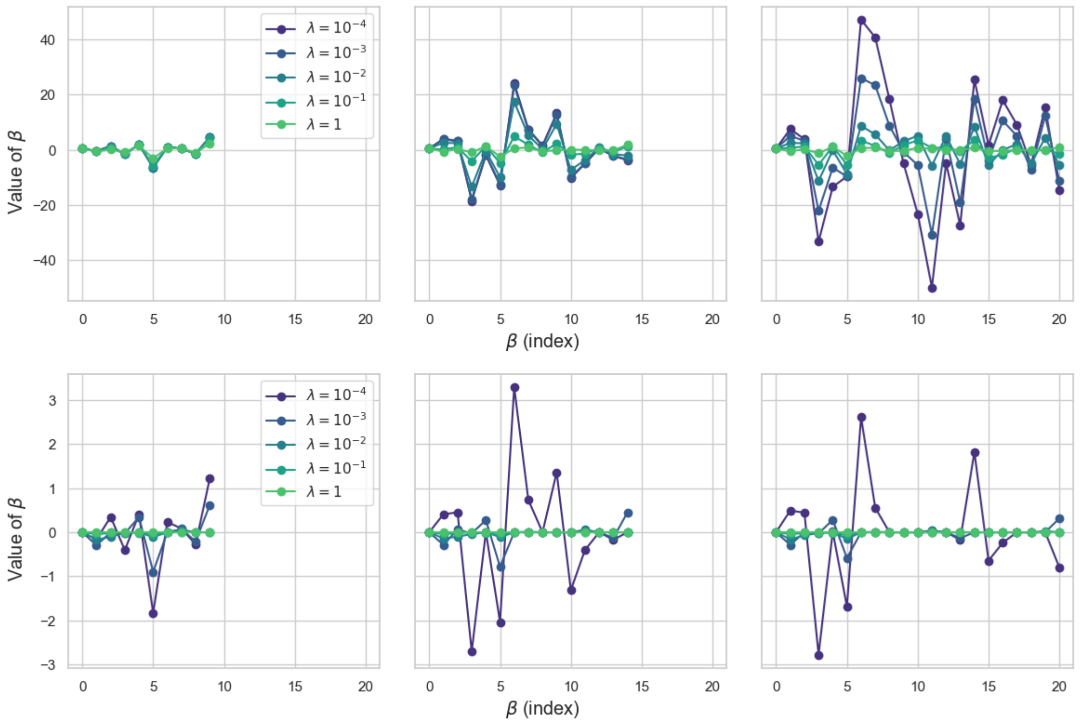

Figure 9. β coefficients for Ridge (above) and Lasso (below) for polynomial degrees three, four and five. We can clearly see the effect of regularization, as the coefficients approach zero when λ increases. Note the difference in scale on the axes and that Lasso-coefficients decrease from degree four to five.

degrees of five or less, the best model by bootstrap-estimated MSE-score was a the degree five model with score 0.0123. This agrees with the 10-fold cross validation results shown in table I. Here, we see that both bootstrapping and cross validation underestimates the true test error for both OLS and Ridge, though the difference is small. On the other hand, cross validation for Lasso regression actually overestimates the error. In testing using fewer folds we found no difference in the estimates, and thus we stuck with 10 folds for the topographical analysis.

Table I. Comparison of OLS, Ridge, and Lasso regression with their respective optimal model. The table presents the optimal regularization parameter λ for Ridge and Lasso, along with the MSE-estimates obtained from cross-validation and the error on the actual test data. OLS and Ridge with low regularization seem equivalent down to differences in data-randomization. Lasso gives slightly higher errors even with the lowest regularization.

Measurement	**OLS**	**Ridge**	**Lasso**
Optimal Degree	5	5	5
Optimal λ		0.0001	0.0001
Cross validation	$1.211 \cdot 10^{-3}$	$1.211 \cdot 10^{-3}$	$1.770 \cdot 10^{-3}$
Test MSE	$1.277 \cdot 10^{-3}$	$1.276 \cdot 10^{-3}$	$1.410 \cdot 10^{-3}$

B. Fitting topographical data

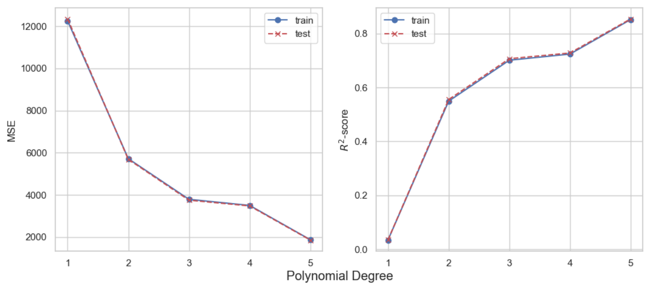

Figure 10. MSE and R^2-score as a function of polynomial degree for OLS applied to topographical data. The left plot shows that the MSE decreases with increasing polynomial degree for both training and test sets, reaching its lowest value at degree five. The right plot shows that the R^2-score increases steadily with model complexity, indicating improved fit to the data with higher polynomial degrees. The close alignment between the training and test curves suggests minimal overfitting.

The low-degree fits on the topographical data reiterates the insights already gained from the Franke function:

1. Figure 10 shows the MSE and R^2-score as a function of polynomial degree for low-degree OLS models. The results are as expected with higher order

models performing the best, and indicate that there is good reason to attempt higher order fits.

2. Results from 10-fold cross-validation for low-degree models with optimised parameters, as shown in Table II, reveal that the MSE values are very similar for OLS and Ridge when λ is small, while Lasso gives almost twice as large an error.

3. In accordance with results from cross-validation, bootstrap resampling results (also in Table II) show little variation between models. OLS has the lowest MSE, while Ridge is only slightly higer and Lasso being almost double the preceding ones.

4. The test MSE values (also in Table II) are higly consistent with the bootstrap results, though Ridge barely comes out on top.

5. Finally, figure 11 gives a visual comparison of the best 5^{th}-degree fits of the OLS, Ridge, and Lasso models. The OLS and Ridge models capture the main features of the terrain effectively. The Lasso model produces a smoothed fit, losing the finer details in the real terrain.

We then move on to higher-order fits, where OLS increasingly outperforms Ridge. We take the low-order results as ample evidence that Lasso is suboptimal for our purposes, giving poor fits and being computationally intensive.

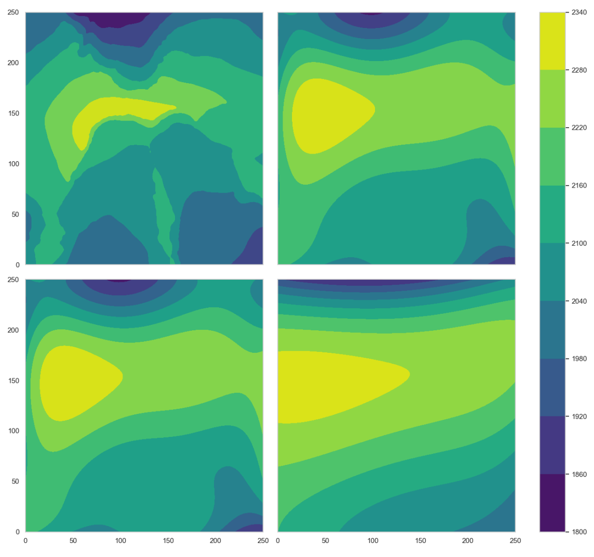

Figure 11. Visual representation of degree five fit for OLS, Ridge, and Lasso compared to the true topographical data. Sequence: the real data (upper-left), OLS fit (upper-right), Ridge fit (bottom-left) and Lasso fit (bottom-right). OLS and Ridge are similar, with Lasso giving a smoother fit.

Table II. Comparison of OLS, Ridge and Lasso models. The table presents the optimal regularization parameter λ for Ridge and Lasso, along with the MSE obtained from cross-validation, bootstrap resampling and test data.

Run	**OLS**	**Ridge**	**Lasso**
Optimal Degree	5	5	5
Optimal λ		0.0001	0.0001
Cross validation	1879.8687	1879.8693	3690.8914
Bootstrap	1862.6967	1862.8328	3654.7688
Test MSE	1862.0418	1862.0295	3653.9606

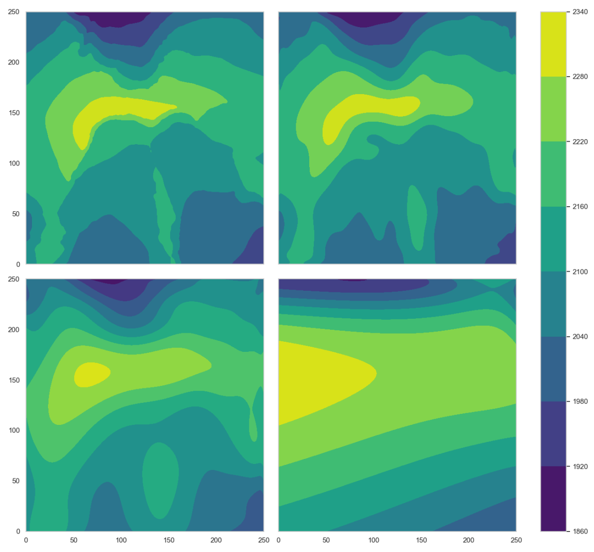

Figure 12. Visual representation of higher-order fits for OLS and Ridge with hyperparameter 10^{-4} and 1, compared with the true topographical data. Sequence: the real data (upper-left), OLS fit (upper-right), fit of Ridge with $\lambda = 10^{-4}$ (bottom-left) and fit of Ridge with $\lambda = 1$ (bottom-right).

We performed tests for degrees up to 30 with OLS and Ridge. Figure 12 provides a visual comparison of the selected models: The best model was OLS at degree 29, and the best ridge model was degree 30 with hyperparameter 10^{-4} A highest-order Ridge fit with $\lambda = 1$ has also been included to demonstrate the smoothing effect of regularization.

Even for small λ-values, the difference between OLS and Ridge is now dramatic. This is demonstrated in figure 13, showing cross validation scores for OLS and Ridge with different λ-parameters for polynomial degrees five to 30. We see that the MSE is quite similar for OLS and Ridge with $\lambda = 10^{-4}$ up to degree seven, but that they diverge after this. OLS consistently has the best MSE-score. Close examination of the figure reveals that OLS actually attains its minimum for degree 29, and starts to increase for higher degrees, and tests up to degree 35 confirm that we reach the best bias-variance trade-off at degree 29.

Finally, figure 14 shows a 3D plot of the actual topographical terrain data and this best model. As is expected from figure 12 and table II, the difference in the two plots are minimal, though the boundary of the model plot exposes the polynomial nature of the fit as it is uanble to flatten completely.

Overall, we found that OLS fits the data best for both the Franke function and the topographical data. For low-degree fits the difference between OLS and Ridge with small λ-values are minor, whereas Lasso performs comparatively worse. For high-degree fits the difference between OLS and Ridge are greater, even for small λ-values.

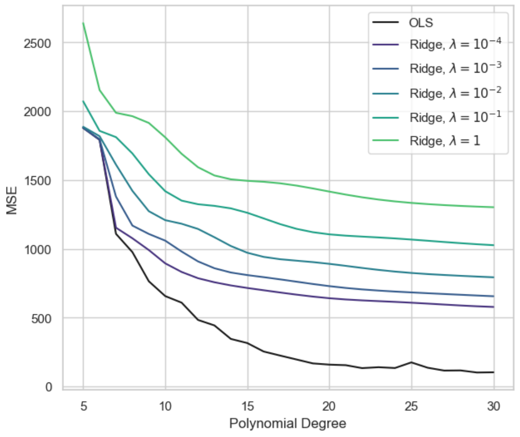

Figure 13. Cross validation scores for OLS and Ridge with different λ-parameters for polynomial degree five to 30. We see that the MSE is quite similar for OLS and Ridge with $\lambda = 10^{-4}$ up to degree seven, but that they diverge after this. OLS consistently has the best MSE-score, and the performance decreases as λ increases.

V. Discussion

1. *Model Comparison*

Comparing OLS, Ridge, and Lasso regression on the Franke function reveals that all three models exhibit near-optimal performance at a polynomial degree of five when the regularization is minimized. This is unsurprising, since the models become near equivalent as $\lambda \to 0$.

Though the margins are small, OLS and Ridge with very small λ-values consistently performs the best on both datasets. The final differences between these methods on the Franke function are so small that they are likely due to the randomization in the data, while the effect is more pronounced on the more complex topographical data, where a $\lambda << 10^{-4}$ would be needed to equalize OLS and Ridge performance for more complex models (figure 13).

This indicates that regularization is unnecessary in our case. To explain when this conclusion may hold more generally, we pick out the following key characteristics of our situation: In both our synthetic and topographical data, we have i) quite a lot of data, ii) limited noise and no outliers, iii) relatively low polynomial degrees and iv) a polynomial design matrix with no risk of multicollinearity. Points i), ii) and iii) together make overfitting very unlikely, and the constant improvement in figure 6 indicates that we make use of the added complexity in our OLS-fits, so that no feature is likely to be superfluous. Thus, it is not surprising that OLS outperforms Ridge, having some regularization, which outperforms Lasso, having the strongest shrinkage. Point iv) establishes that there is no good reason to prefer Ridge over OLS for stability reasons in our case.

With higher regularization terms, the performance of both Ridge and Lasso suffers, suggesting that strong regularization excessively penalizes model complexity, leading to underfitting. However, figure 8 clearly demonstrates a key difference between these models; the polynomial degree is the primary predictor of MSE for Ridge, while the regularization term is the main predictor for Lasso. Particularly, for simpler Ridge models (e.g. degree one), the MSE remains stable regardless of λ, reflecting that Ridge regularization has a limited impact on less complex models with fewer parameters to penalize. For Lasso, even the simple models show variance between different regularization parameters as they are forced all the way to zero. Thus, a Lasso fit with $\lambda = 1$ loses virtually all information and becomes equivalent for all tested polynomial degrees.

The main drawback of Lasso regression is the lack of an analytical solution. We find that Lasso is dramatically slower to run than OLS and Ridge, especially in combination with k-fold-optimization or bootstrap analysis. Therefore, our tests on the synthetic data indicate that Lasso is a bad choice for our complex topographical dataset. This was confirmed on this dataset by basic tests up to degree five, and which is why we focused our further analysis on OLS and Ridge. For a surface fitting task with sparse data, where there is risk of overfitting with OLS, we consequently believe that Ridge should generally be preferred to Lasso. Still, the case for using Lasso can be made if feature selection is important, since only Lasso will completely zero out the unimportant features.

Overall, we conclude that OLS is probably the best option for datasets where conditions i-iv) are satisfied. In the introduction, we listed potential applications such as predicting general terrain features from sparse datasets or beyond the boundaries of a given dataset, particularly in mapping surfaces such as the sea floor or other planets, where data is expensive. For this application, we may be more interested in the underlying, smooth structures of the landscape, and regularization could be useful.

2. *Topographical data*

For the topographical data, we see many of the same tendencies for low-degree fits as we did with the data from the Franke function. This is to be expected since the situations are similar, both satisfying conditions i-iv).

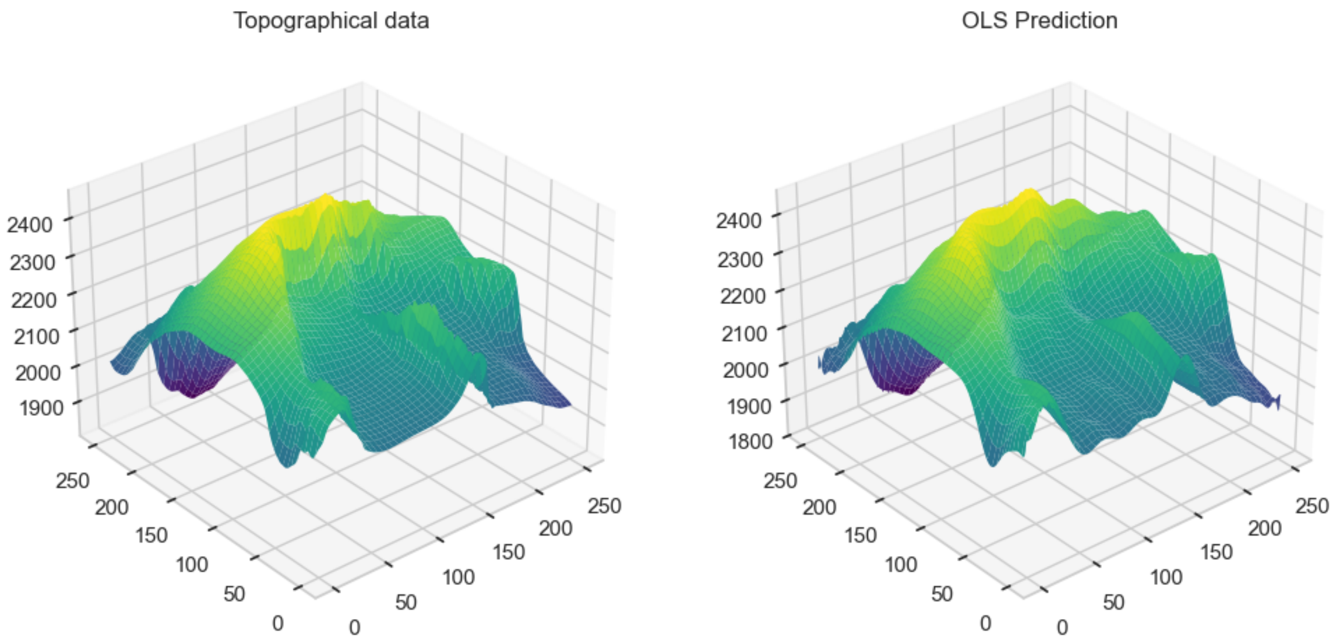

Figure 14. 3D visualization comparing the our best model, OLS of degree 29 to the actual topographical terrain.

Importantly, the scales are different, which explains why the MSE is much higher in all topographical cases; here, the data varies between around 1,800m to 2,400m, and so it is not surprising that the MSE hovers around 12,000 for a polynomial fit of degree one, which is essentially a plane. A plane parallel to the xy-plane at the median height of 2,100m would have deviations of up to 300m at certain points.

Additionally, we see that cross validation and bootstrapping overestimates the error, whereas it actually underestimated the error for the fit of the Franke function for OLS and Ridge. One might perhaps think that this was due to us using analytical expressions and different scaling (only centering) for these two models, whereas all models implemented with `scikit-learn` and `StandardScaler` had their errors overestimated, but we actually made an additional implementation of OLS and Ridge with `scikit-learn` and used these models for the bootstrapping and cross-validation to ensure a fair comparison. We are therefore unsure as to the cause of this effect, though it is not very dramatic (table II). In the end, we see that the bootstrap method quite accurately predicts the test error, while cross validation slightly overestimates it.

Our studies of the OLS fit for high polynomial degrees shows that the MSE starts to increase for polynomial degrees above 29. This indicates that for degrees greater than 29 we are starting to overfit our data, so that we have found the optimal mode with respect to the bias-variance trade-off. When also taking into consideration the computational costs, figure 13 indicates that a degree 20 polynomial is near-optimal. This result suggests that for fitting rugged terrain such as a mountains, a slightly higher-degree fit (at least 15) is worth the extra computational costs for the rapidly improving accuracy, since the variance is of a much smaller scale than the bias given many data points, as indicated by our synthetic data experiments (figure 4).

3. Future Research and Limitations

Future research could explore the applicability of these models, especially the high-degree models, to more general terrain, or prediction of terrain beyond the training boundary. In such a case, one would need the model to pick up on broader trends, aiming to find a smoother fit which will generalize better. In such cases, the regularization of Ridge and Lasso might prove beneficial, helping the model to look beyond local fluctuations. Such predictions would increase the effective area mapped per data point in challenging conditions such as the deep sea or other planets. With this same goal in mind, one could also explore how the models fare with sparser datasets.

Due to the limited scope of this project, it was not deemed worthwhile to make proper class implementations of the regression methods, since their application is already relatively seamless with the `Pipeline`-functionality in `scikit-learn` [1]. However, if future studies expand on our work, this may be a natural improvement to make.

VI. Conclusion

We have explored regression methods applied to both synthetic data (Franke's function) and real-world topographical data from Glittertind, Norway. The main objective was to find the best model balancing computational cost and goodness-of-fit. With this in mind, it was apparent that Lasso was not a good choice for our dataset, being very slow due to numerical convergence and with strong regularization, which leads to underfitting on our complex data. We therefore focused our further analysis on OLS and Ridge.

For low-degree fits, both OLS and Ridge with minimal regularization ($\lambda \approx 10^{-4}$) have similar performance. Still, since our models use polynomial design matrices, there is no risk of multicollinearity and we see no good reason to use Ridge. Increasing the model complexity makes it apparent that OLS is the better choice, and figure 14 demonstrates its accuracy visually.

Overall, we demonstrate that for complex datasets such as topographical data of Glittertind, with i) quite a lot of data, ii) limited noise and no outliers, iii) relatively low polynomial degrees and iv) a polynomial design matrix with no risk of multicollinearity, OLS outperforms Ridge and Lasso. In our specific case, a degree 20 polynomial OLS model provides a near-optimal balance of limiting both MSE and computational costs.

[1] 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, Journal of Machine Learning Research **12**, 2825 (2011).

[2] 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, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Nature **585**, 357 (2020).

[3] R. T. Trevor Hastie and J. Friedman, *The Elements of Statistical Learning* (Springer New York, NY, 2009).

[4] M. Hjorth-Jensen, "Applied Data Analysis and Machine Learning," `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter3.html` (2021).

[5] Y. H. L. Sebastian Raschka and V. Mirjalili, *Machine Learning with PyTorch and Scikit-Learn* (Packt Publishing, Birmingham, UK, 2022).