FYS-STK3155 Project 1

Lasse Pladsen, Parham Qanbari, & Sander V. Vattøy

October 13, 2023

Abstract

We have studied three different regression methods, Ordinary Least Squares (OLS), Ridge, and Lasso, and what affects their learned prediction accuracy. Using real world terrain data of the Oslo region, we found that OLS provided the most accurate prediction with the lowest MSE of 178986.87. The OLS model with polynomial degree 17 provided the lowest MSE. For Ridge regression we instead found the optimal degree to be 18 with $\lambda = 10^{-4}$. Using Lasso regression we found that the MSE of our model decreased by about 9% with optimal degree 17. We have shown the concept of overfitting using the Franke function, and demonstrated models' bias-variance trade-off. We then study the cross-validation resampling method and analyze all three regression methods. We observed that increasing the fold amount k from 5 to 10 decreases the MSE by a magnitude of 10^4 of all models.

I. INTRODUCTION

In today's modern world machine learning has emerged as a revolutionary technology. Even if just superficially, machine learning has become well known around the world. By allowing computers to learn from supplied data we create powerful tools with incredible real world applications for analyzation and prediction.

In this project we will explore and study low-level machine learning methods. Our focus will be on understanding and optimizing the potential of data regression methods such as ordinary least squares (OLS), Ridge regression, and Lasso regression. We will look into the factors that influence their accuracy and predictive capabilities, employing a variety of strategies to minimize errors and improve performance. Thereafter we will dive into two resampling methods called bootstrap and cross-validation, to see how this affects our model. Finally, we will apply our finely-tuned algorithms to analyze real-world topographic data over Oslo, Norway.

II. THEORY

A. Design matrix for linear regression

We create a so-called design matrix $\mathbf{X}$ for the regression methods from two input variables $\mathbf{x}$ and $\mathbf{y}$. This matrix is used to create our linear regression models. Each row in $\mathbf{X}$ represents polynomial variables from one data sample. We choose a max polynomial degree p with n data samples such that $\mathbf{X} \in \mathbb{R}^{n \times l}$ where

$$l = floor[\frac{1}{2}(p+1)(p+2)],$$

where the $floor$-function rounds a value down to the nearest integer, and the design matrix is then given by (Hjorth-Jensen, 2023)

$$\mathbf{x} = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0 y_0 & y_0^2 & \dots & x_0^{p-1} & \dots & y_0^{p-1} \\ 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 & \dots & x_1^{p-1} & \dots & y_1^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & y_{n-1} & \dots & \dots & \dots & \dots & x_{n-1}^{p-1} & \dots & y_{n-1}^{p-1} \end{bmatrix} \tag{1}$$

We then make our linear regression predictions by creating the polynomials from $\mathbf{X}$ and the polynomial coefficients $\boldsymbol{\beta} \in \mathbb{R}^l$ as follows (Hjorth-Jensen, 2023)

$$\tilde{\mathbf{z}} = \boldsymbol{X}\boldsymbol{\beta} \tag{2}$$

B. Regression methods

The different methods have different ways of calculating the optimal $\boldsymbol{\beta}$-coefficients which we call $\hat{\boldsymbol{\beta}}$. The following expressions are derived in the course lecture notes (Hjorth-Jensen, 2023) unless otherwise noted.

B1. Ordinary least squares

In the method ordinary least squares we find our optimal coefficients $\hat{\boldsymbol{\beta}}$ by minimizing the squared difference between $\mathbf{z}$ and $\tilde{\mathbf{z}}$:

$$\frac{1}{n}(\mathbf{z}_i - \tilde{\mathbf{z}}_i)^2 = \frac{1}{n}||\mathbf{z} - \mathbf{X}\boldsymbol{\beta}||_2^2$$

where we use the norm-2 definition for a vector

$$||\mathbf{x}||_\mathbf{2} = \sqrt{\sum_\mathbf{i} \mathbf{x_i^2}}$$

. We then attain the following expression

$$\hat{\boldsymbol{\beta}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z} \tag{3}$$

B2. Ridge

For the Ridge regression method we add a regularization parameter λ such that we minimize the following

expression

$$\frac{1}{n}||\mathbf{z}-\mathbf{X}\boldsymbol{\beta}||_2^2+\lambda||x_i||_1$$

This leads to the following expression

$$\hat{\boldsymbol{\beta}}_{Ridge}=\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{z} \tag{4}$$

where $\mathbf{I}$ is the identity matrix. In addition the parameter must be non-negative $\lambda \geq 0$.

B3. Lasso

Lasso regression stands for 'least absolute shrinkage and selection operator' where we instead want to minimize

$$\frac{1}{n}||\mathbf{z}-\mathbf{X}\boldsymbol{\beta}||_2^2+\lambda||x_i||_1 \tag{5}$$

where we here also use the the norm-1 definition for a vector

$$||\mathbf{x}||_\mathbf{1}=\sum_\mathbf{i}|\mathbf{x_i}|$$

We do not have an analytical expression for $\boldsymbol{\beta}_{Lasso}$ so this needs to be numerically calculated.

C. Standardization scaling

For machine learning algorithms in general it is normal to scale our data to avoid extreme values affecting the analysis of the results. This can help improve the model's performance. One such frequently used scaling method is standardization, which we are going to use. This method makes sure each feature (column) of the design matrix (1) has a mean value of zero and standard deviation of one. Each feature x_j is then scaled by the following expression

$$x_j \to \frac{x_j-\bar{x}_j}{\sigma(x_j)} \tag{6}$$

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

D. Measuring error and coefficient of determination

To calculate our regression models' prediction error from the actual data we will be using the mean square error (MSE) defined by

$$MSE(\mathbf{z},\tilde{\mathbf{z}}_i)=\frac{1}{n}\sum_{i=0}^{n-1}(z_i-\tilde{z}_i)^2, \tag{7}$$

The coefficient of determination, denoted R^2, is a measure that tells us how well our models can predict new data and is defined as

$$R^2(\mathbf{z},\tilde{\mathbf{z}}_i)=1-\frac{\sum_{i=0}^{n-1}(z_i-\tilde{z}_i)^2}{\sum_{i=0}^{n-1}(z_i-\bar{z})^2} \tag{8}$$

where $\bar{z}$ is the mean value of $\mathbf{z}$, and a value of $R^2=1$ will represent the best possible model determination from our regression method. A value of one means the model's prediction is perfect, and a value of negative one means the model's prediction can not predict any of the data.

E. Resampling

Resampling is to repeatedly gather samples from the same data set to increase a model's performance or to possibly obtain extra information which otherwise would not be possible with only one sample/data split. When it comes to the linear regression methods of this project we can resample our data's training split then do a regression fit on each sample. By using the average we can hopefully the model's accuracy. The two resampling techniques we are going to use in this project is bootstrap and k-fold cross-validation.

F. Bootstrap

The bootstrap resampling technique uses random sample selection from the data set n number of iterations with replacement. Because we replace each data sample before every iteration some data may be sampled more than once and some may never be sampled.

G. k-fold cross-validation

The k-fold cross-validation resampling technique is where the data set gets split into k number of equally sized subsets (folds) containing the data. A single fold is reserved for testing the model, and the rest $k-1$ number of folds is then used to train the model. This is repeated k times where every iteration a different fold is used for the testing and the rest for training (Hjorth-Jensen, 2023). In this project we make our own code for k-fold by using `KFold` from `sklearn.model_selection` (inspired from (codebasics, 2019)).

H. Bias and variance

Expressing (7) as the expectation value

$$MSE=\mathbb{E}[(\mathbf{z}-\tilde{\mathbf{z}})^2]$$

we can rewrite this as

$$MSE=\text{Bias}[\tilde{\mathbf{z}}]+\text{var}[\tilde{\mathbf{z}}]+\sigma^2 \tag{9}$$

where

$$\text{Bias}\left[\tilde{\mathbf{z}}\right]=\mathbb{E}\left[\left(\mathbf{z}-\mathbb{E}\left[\tilde{\mathbf{z}}\right]\right)^2\right] \tag{10}$$

and

$$\text{var}\left[\tilde{\mathbf{z}}\right]=\mathbb{E}\left[\left(\tilde{\mathbf{z}}-\mathbb{E}\left[\tilde{\mathbf{z}}\right]\right)^2\right]=\frac{1}{n}\sum_i(\tilde{z}_i-\mathbb{E}\left[\tilde{\mathbf{z}}\right])^2 \tag{11}$$

Here σ is the standard deviation of our stochastic noise $\boldsymbol{\epsilon}$. The derivations of (9) can be found in the appendix section C.

Here we see that the MSE is just the sum of the bias and the variance of the model, in addition to the σ^2 term (the noise variance). Figure 1 shows how the idealized regions of low vs high bias and variance im comparision to the MSE. The figure also shows an example of overfitting. This is where we fit the model too close to the training data which results in less prediction performance for unseen data like the testing data set.

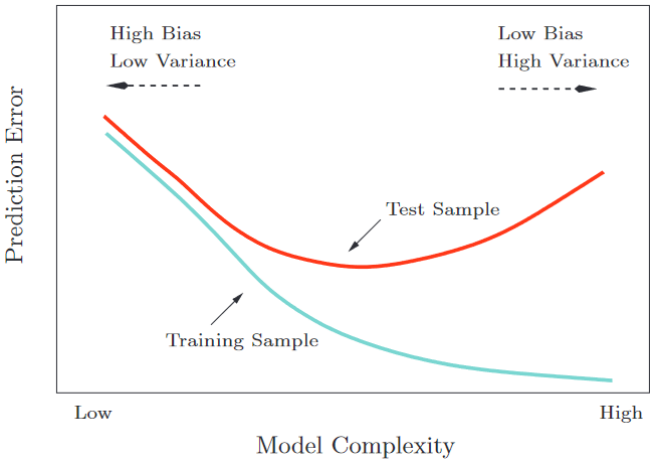

Figure 1: Figure 2.11 from Hastie et al. (2009). This idealized graph shows the bias-variance trade-off for our models, and shows the concept of overfitting model to the training data. This happens when we use too high complexity for the model leading to higher errors and worse prediction capabilities.

I. The Franke function

To study our regression methods we will be using the Franke function, which is a two dimensional weighted sum of four exponentials given as

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left[-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right] \\ &+ \frac{3}{4}\exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right] \\ &+ \frac{1}{2}\exp\left[-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right] \\ &- \frac{1}{5}\exp\left[-(9x-4)^2 - (9y-7)^2\right] \end{aligned} \tag{12}$$

III. METHODS

In this project we will both write our own code in Python and take use of many functionalities of the machine learning python module `SciKit-Learn`. For every step we are going to scale our design matrix by standardization (6) using `SciKit-Learn`'s `sklearn.preprocessing.StandardScaler`. For each step we will also split our data into a training set and a testing set using `Scikit-Learn`'s `model_selection.train_test_split` function.

Our first step is to study how the MSE and R^2 changes as the model complexity changes using our three regression methods. For this we will vary the complexity by changing the maximum polynomial degree p in our design matrix $\mathbf{X}$.

For our data set will be using both the Franke function (12) and real terrain data of Oslo with $x_i, y_i \in [0,1]$ with N data points. For the Franke function we simulate a stochastic noise from a normal distribution $\epsilon \sim \mathcal{N}(0,1)$. For the terrain data we will use take a square section with size $N \times N$, we will specifically use $N = 500$ for all the terrain analysis. This means we gather the chosen terrain section from the slice of the N first x and N first y values (`terrain_section = terrain[:N, :N]` in Python).

Firstly we will do the OLS regression, writing our own code from (3), then we will do a Ridge regression from (4) with varying λ-parameter, to find the error in each case. Thirdly we will use `Scikit-Learn`'s implementation of Lasso regression (`sklearn.linear_model.Lasso`). For each of these regression methods we will study the prediction accuracy with respect to varying maximum polynomial degree p.

Next up we will be introducing bootstrap resampling where we use `Scikit-Learn`'s implementation `sklearn.utils.resample` with $N_{bootstrap}$. Thereafter we do a bias-variance trade-off analysis of the OLS regression, and recreate the theoretical low vs high bias and variance. From this we will find how the mean square error changes with respect to polynomial degree.

Now we create a code for the cross-validation resampling method, where we use `Scikit-Learn`'s implementation `sklearn.model_selection.KFold`. Here we will study the MSE of all three regression methods using cross-validation with different fold parameters $k \in [5, 10]$, and thereafter compare the error with the result found using the bootstrap method.

We apply these methods and train the regression models OLS, Ridge and Lasso were we get a output $\hat{\boldsymbol{\beta}}$ which is our coefficients we will use to make a prediction of the terrain data. Firstly, we use $\hat{\boldsymbol{\beta}}$ to model the terrain which the model is already trained on for two reasons. One, to see which features of the terrain the model learns to predict, more specifically, we want to see if the model is able to predict extreme peaks and deep valleys. Secondly, how well the model is able to recreate the terrain by comparing MSE between real terrain and modeled terrain. We then use the model to predict a completely new terrain in a attempt to see how well the model performs on a dataset it has not been trained on. Essentially to measure the models generalizability.

IV. RESULTS

The analysis results from the Franke function can be found in appendix D, but here we focus on the the analysis on the real terrain data of Oslo.

A. Ordinary Least Squares

Using OLS regression we trained our model on the chosen 500×500 area of the total terrain. The model on the same area is plotted in 2, and on a new unused area is plotted in figure 3. The MSE and R2 as functions of polynomial degree are respectively plotted in figure 4 and figure 5. We see the MSE values clearly dropping as the model complexity increases (about 13% over 17 polynomial degrees), while the R2 score have negative value, and decrease in absolute value as complexity increases (goes from around -6.5 to -5.5). The β parameters are plotted in figure 6, where greater model complexity give us smaller values for the lower β-parameters. For example for $degree = 0$ we have $\beta_{30} \simeq 2.0 \cdot 10^7$, while for $degree = 19$ $\beta_{30} \simeq 0.3 \cdot 10^7$. For higher β-parameters we can on the contrary see that lower degrees have very small β-values, compared to the more stable β-variance for higher degrees.

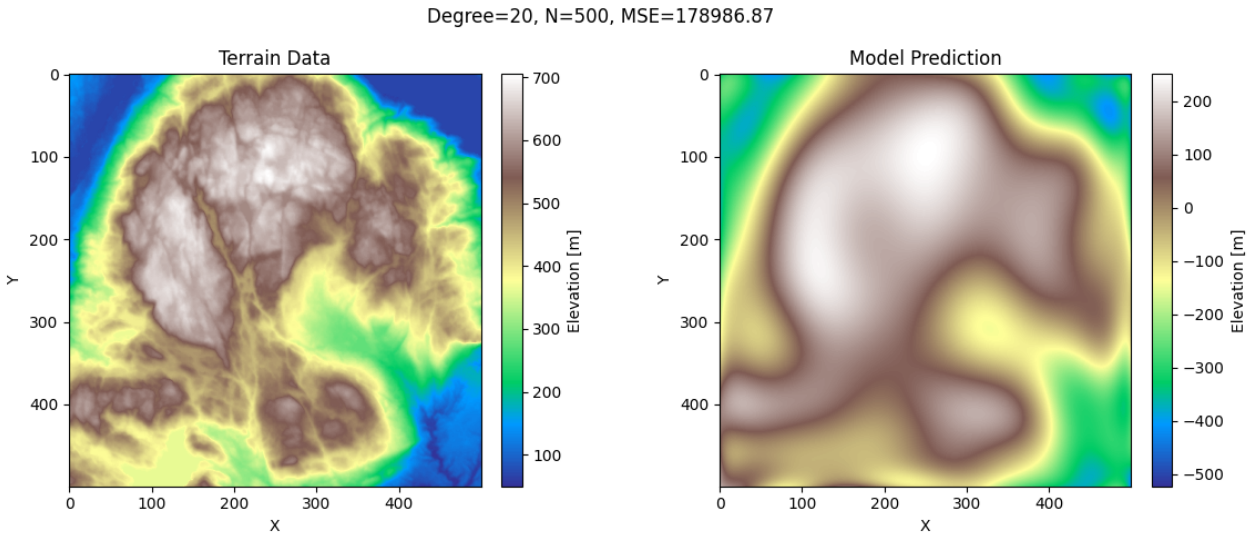

Figure 2: Plot of the model prediction on the same area that was used for training.

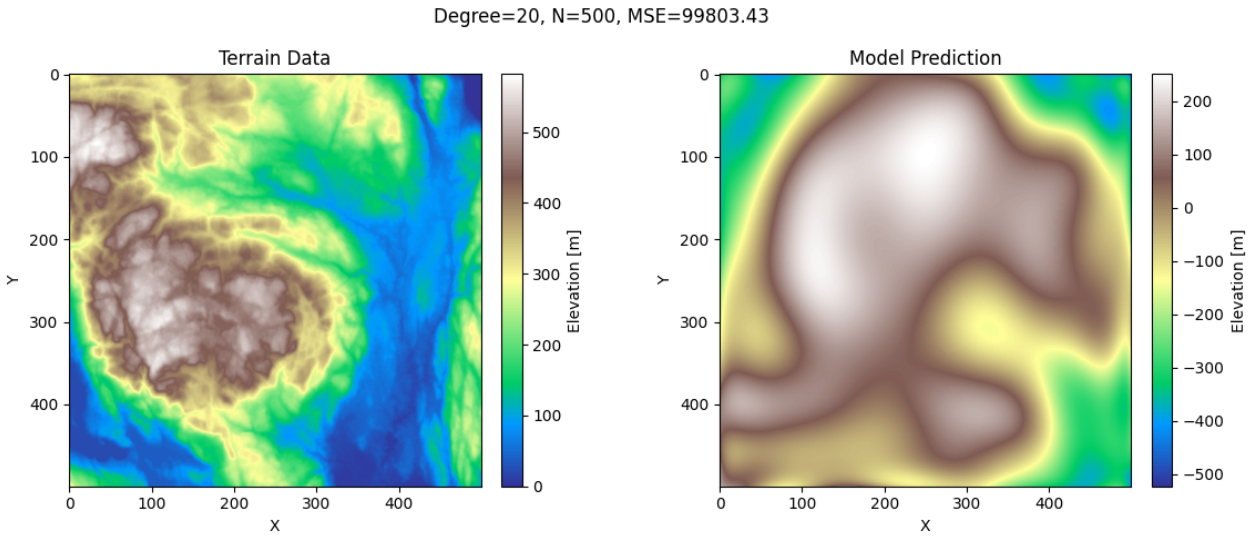

Figure 3: Plot of the model prediction on a new area.

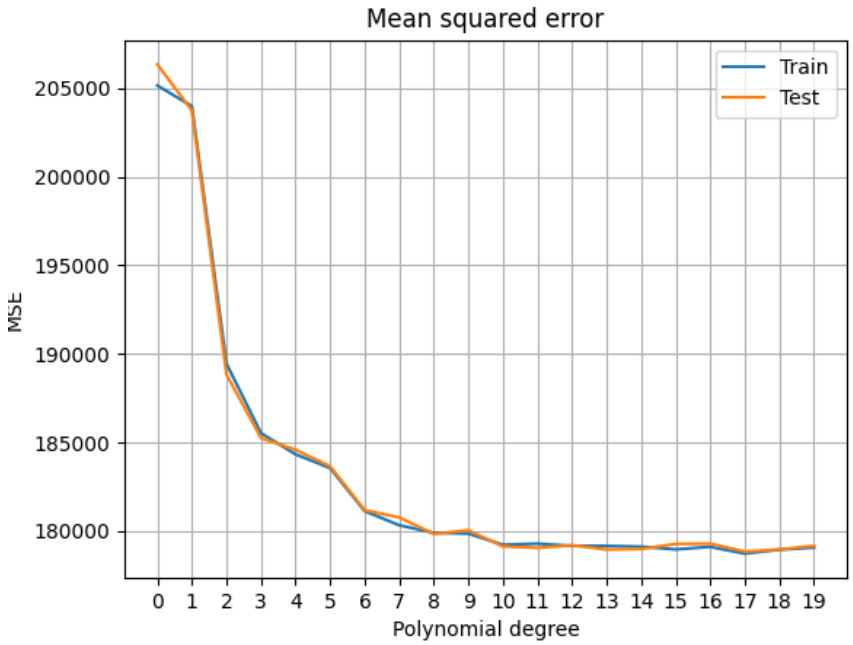

Figure 4: Graph of the MSE of the OLS model on the used terrain area.

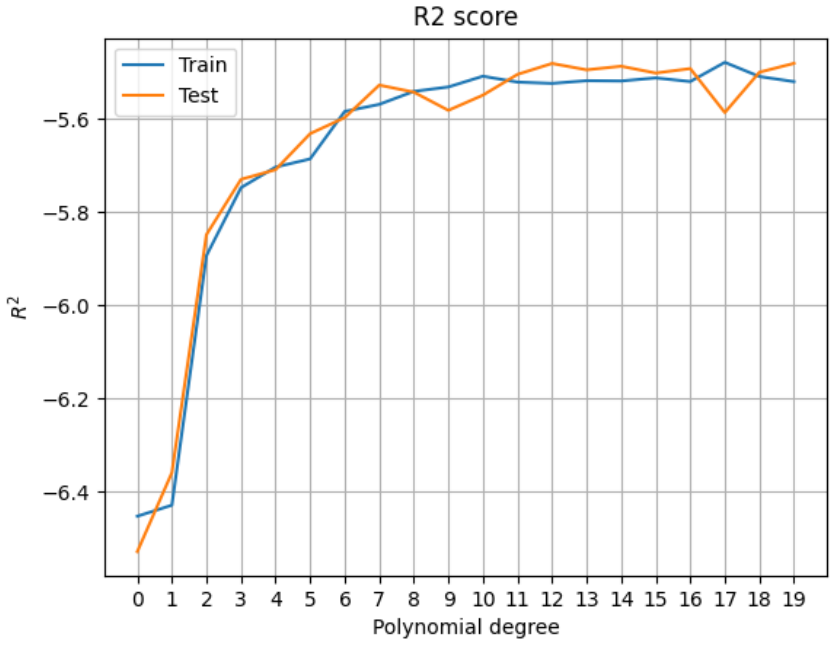

Figure 5: Graph of the R2 scores for the OLS model on the used terrain area.

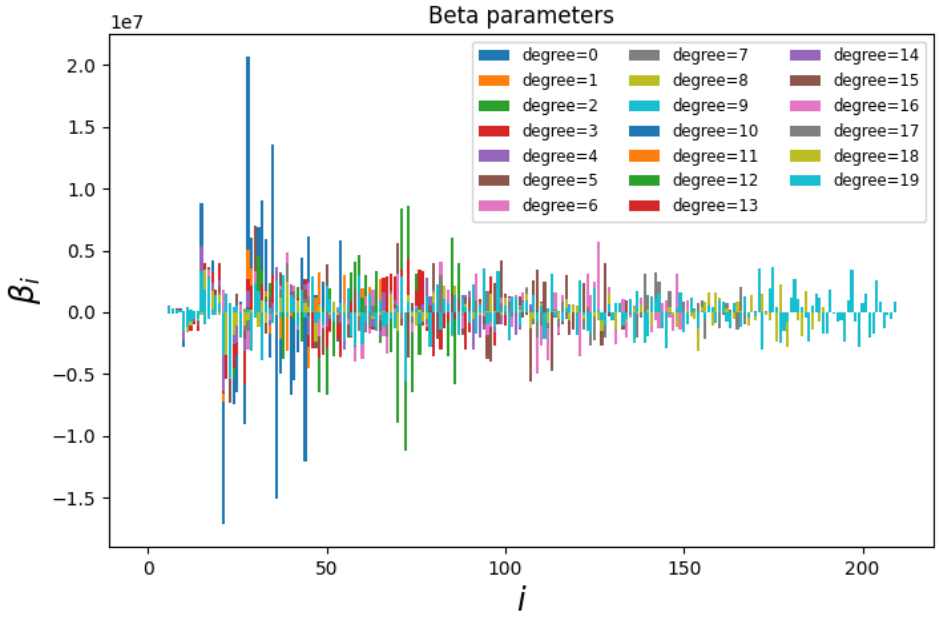

Figure 6: Plot of the beta parameters from the OLS model.

B. Ridge

In the next step we trained the model using Ridge regression. Figure 7 plots the model, where it is possible to see good relations between terrain and model prediction. Figure 8 plots the MSE as function of the hyperparameter λ, where higher polynomial degree result in lower MSE values for the variable λ values. Lowest MSE is around 179 000 at degree 18 with $\lambda = 10^{-4}$. At last figure 9 plots R2 as function of λ. Again we see that higher degree relates to R^2 value closer to the optimal $R^2 = 1$. Best R^2 value equals around -5.48 at degree 19 and $\lambda = 10^{-4}$.

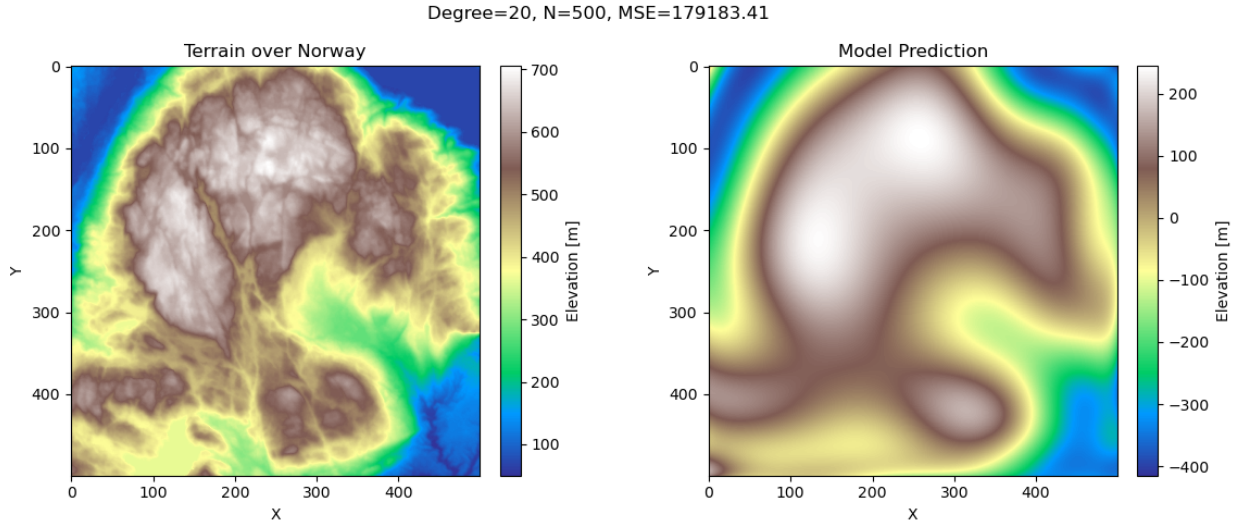

Figure 7: The Ridge regression model using the optimal lambda value that yielded the lowest MSE - $\lambda = 0.0001$.

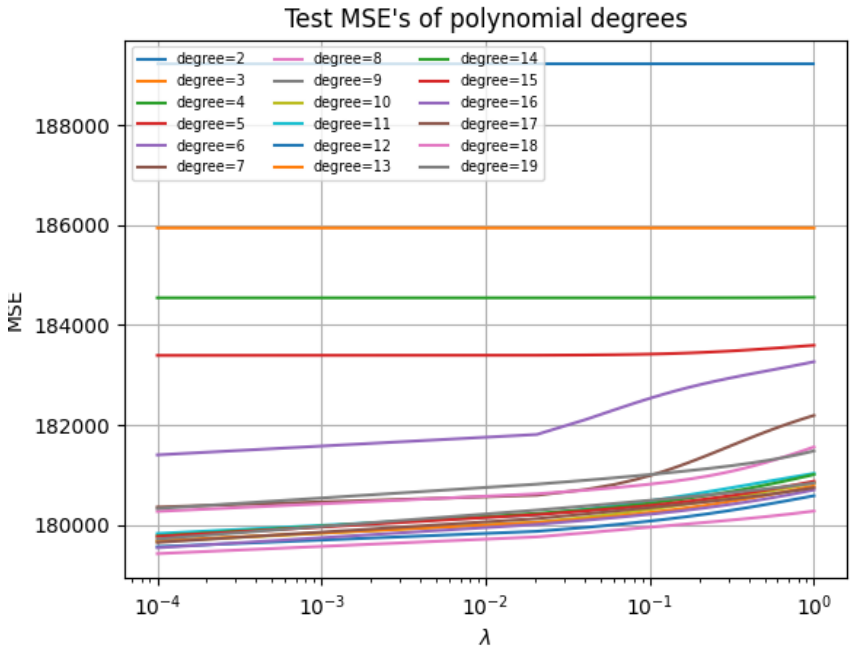

Figure 8: Plotting the test MSE values of the model for each degree as a function of λ values.

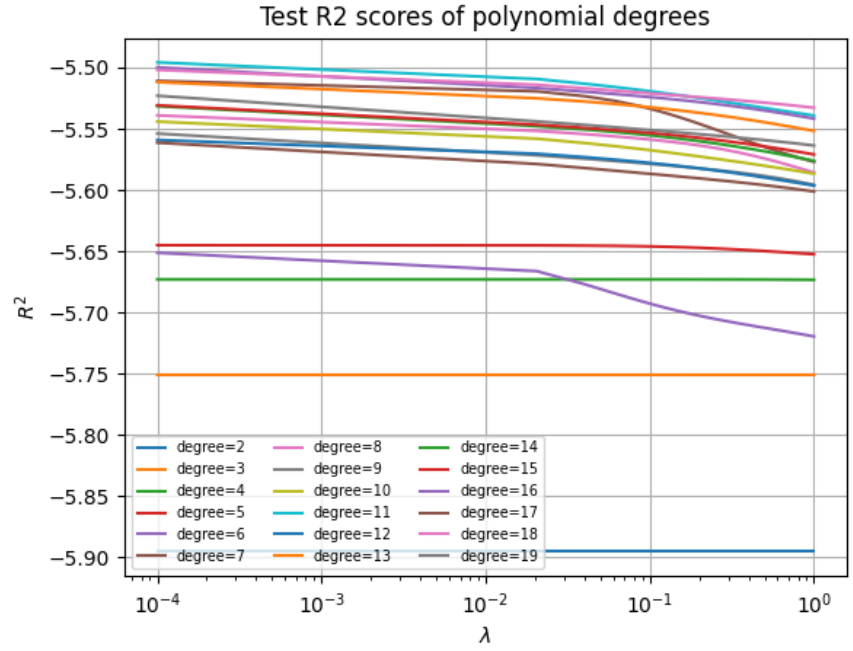

Figure 9: Plotting the test R2 values of the model for each degree as a function of λ values.

C. Lasso regression

We implement the Lasso regression for training our model, the model is plotted in figure 10, and the model prediction seem very bad. The MSE and R2 as functions of polynomial degree are plotted respectively in figure 11 and figure 12. MSE gets reduced with increasing polynomial degree (reduced by about 9% over 17 polynomial degrees). As for the R^2 score, it goes toward the optimal value for increasing polynomial degree (from around -6.45 to -5.78).

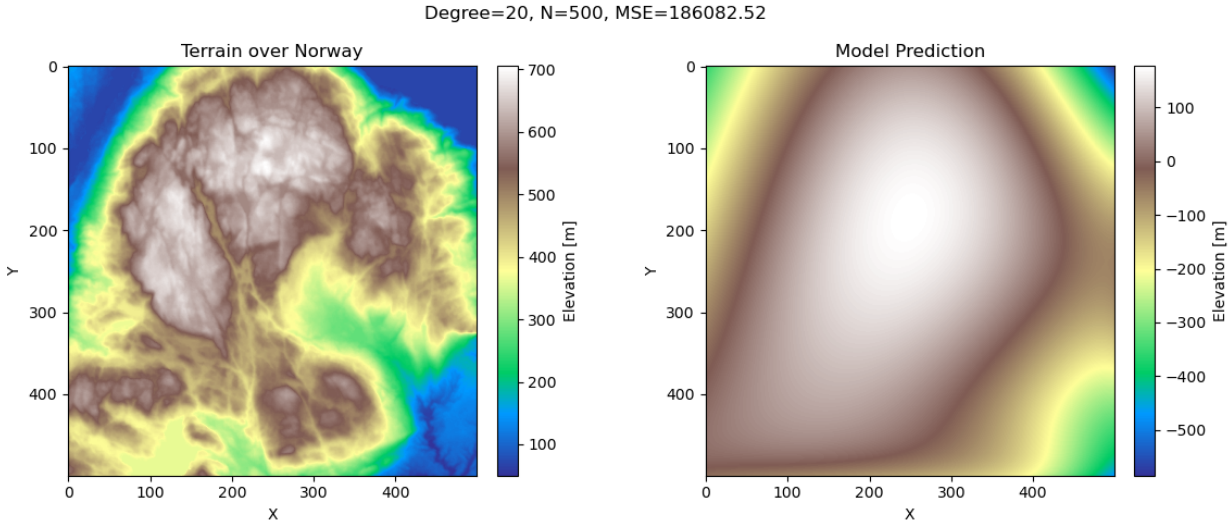

Figure 10: The model trained using Lasso regression plotted on the used terrain area.

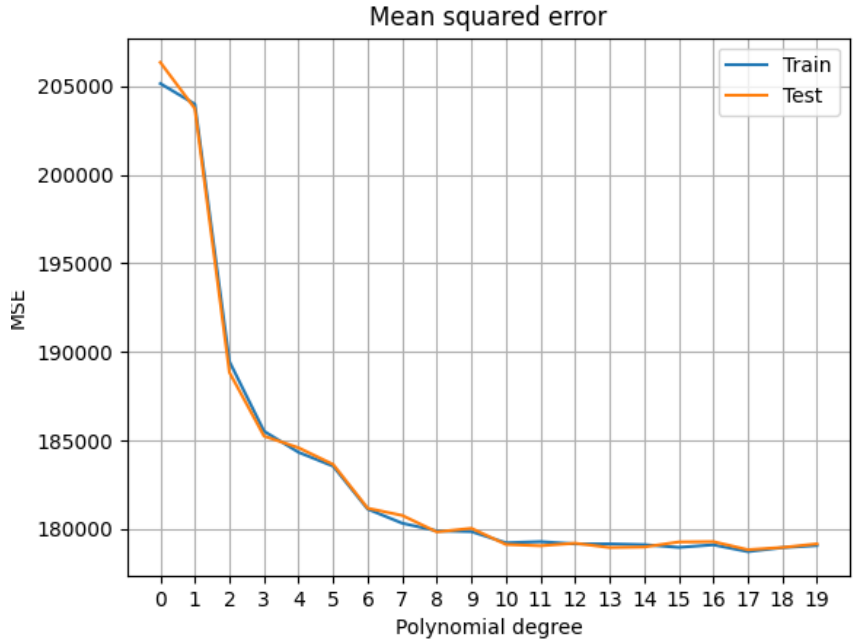

Figure 11: Plotting the test-train MSE values of the Lasso regression model for each degree as a function of polynomial degrees.

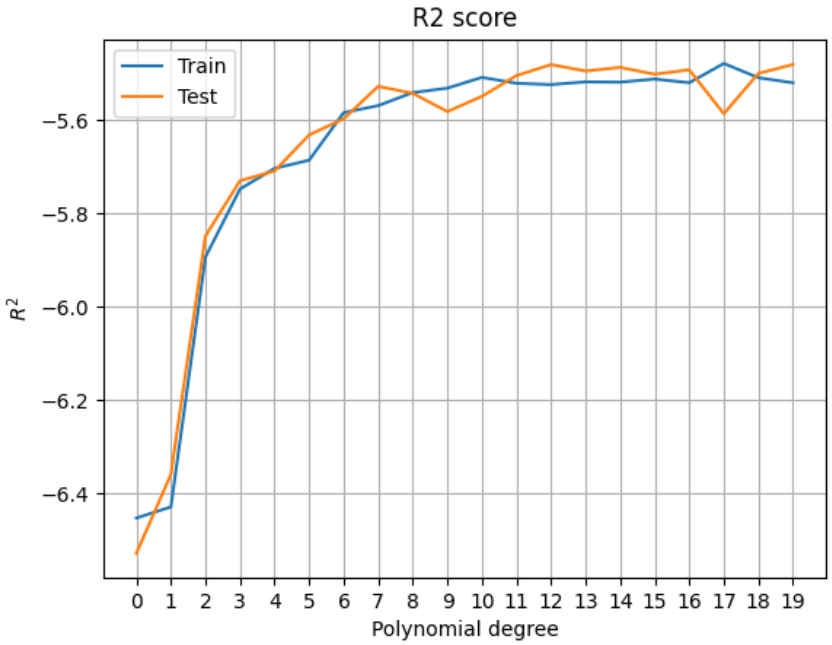

Figure 12: Plotting the test-train R2 values of Lasso regression model for each degree as a function of polynomial degrees.

D. Bias and Variance Analysis

The bias-variance trade-off was too computationally inefficient to run using the terrain data. Therefore, we base our observations on the results from the Franke function. Figure 13 demonstrates overfitting, and figure 14 demonstrates the bias-variance trade-off. Training and test data diverge due to overfitting, and as complexity increases we go from high bias and low variance, to low bias and high variance as described in figure 1.

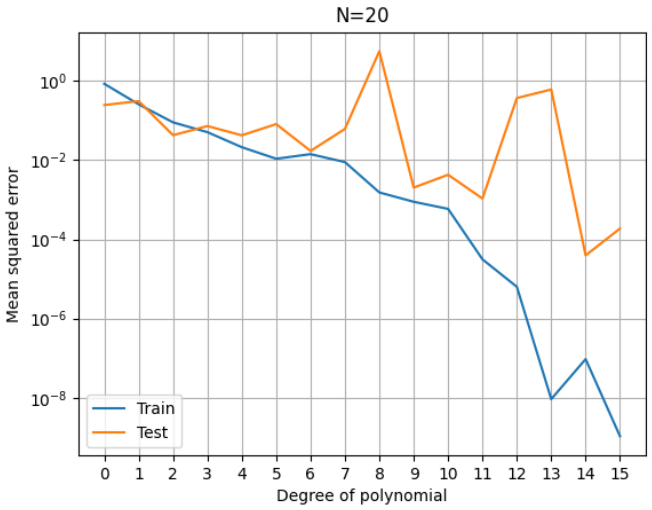

Figure 13: Training and testing MSE as function of polynomial degree using Franke function. This demonstrates overfitting when the complexity is increased too much. Essentially a reproduction of figure 2.11 of Hastie et al. (2009) shown in figure 1.

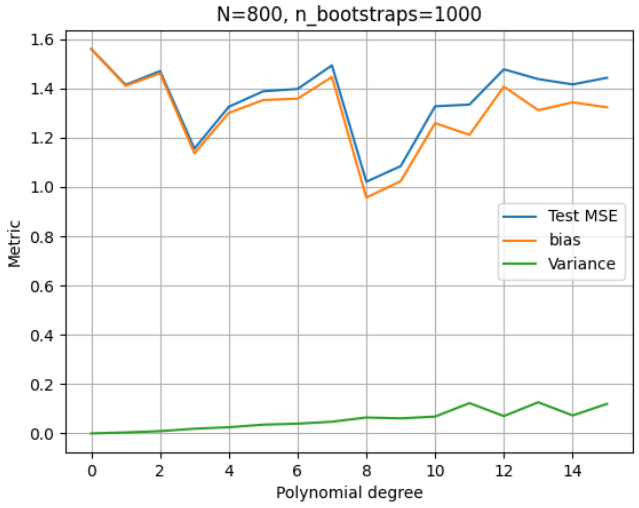

Figure 14: Test MSE, bias, and variance plotted as functions of polynomial degree using the Franke function. This somewhat shows bias-variance trade-off. The bias decreases and variance increases as the complexity increases, as shown ideally in figure 1.

E. Cross validation

Implementing k-fold cross validation for OLS, Ridge and Lasso regression, we plot the MSE as functions of polynomial degree for $k = 5$ folds in figure 15, and for $k = 10$ folds in figure 16. Thereafter we ran the Lasso regression model with 10 folds to predicting the terrain which resulted in figure 17

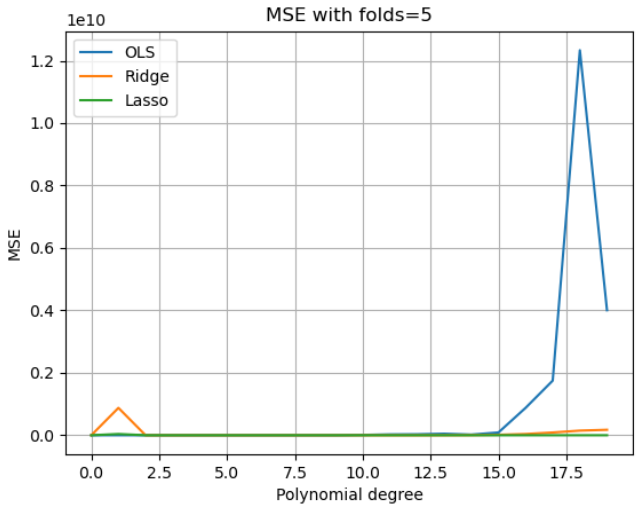

Figure 15: Plots MSE as function of polynomial degree of OLS, Ridge, and Lasso regression using cross validation with 5 folds.

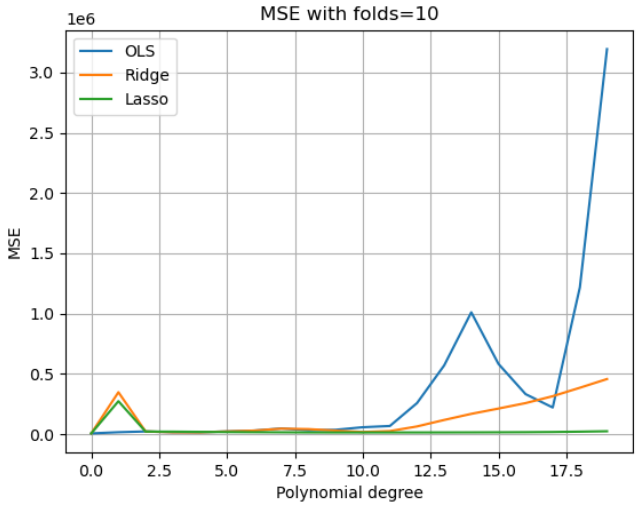

Figure 16: Plots MSE as function of polynomial degree of OLS, Ridge and Lasso regression using cross validation with 10 folds.

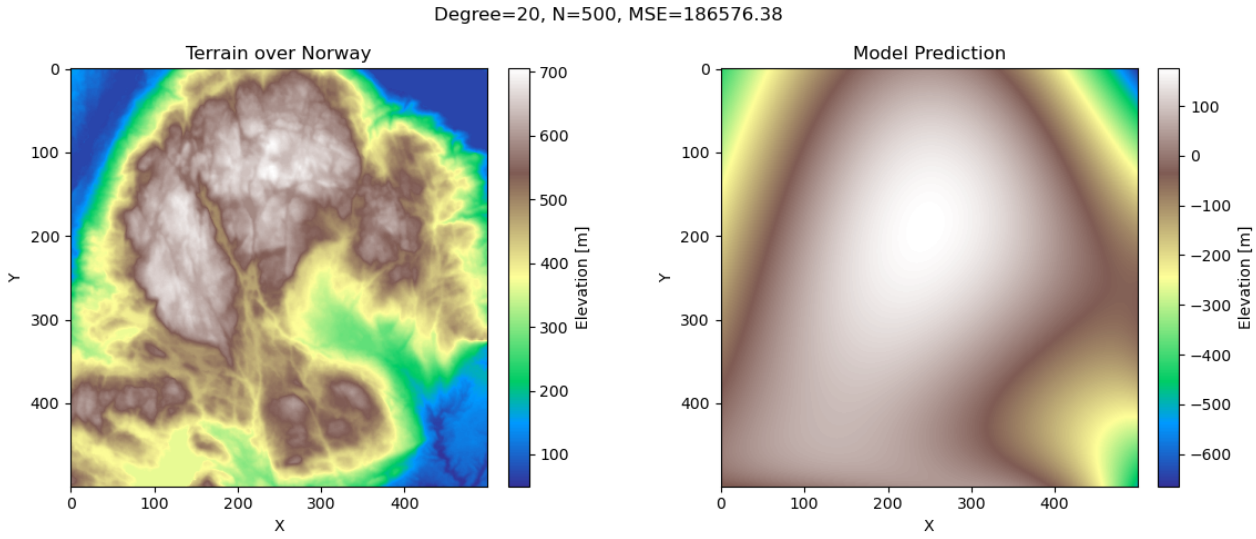

Figure 17: Lasso prediction with 10 folds.

V. DISCUSSION

A. Ordinary Least Squares

When it comes to the beta parameters (figure 6), in general we have higher beta-values for lower model complexity, and lower beta-values for higher model complexity. This show us that lower model complexity have higher variance, which contradicts the theory described in figure 1 (should go from low- to high variance). Still, the lowest degrees have such low values that they do not even show up in the plot meaning they have even lower variance than the higher degrees.

The choice of terrain has caused our model to have an high MSE as observed in figure 4 where the model was applied to the same area it was trained on. A qualitative observation is that the OLS model is able to reasonably model borders between alleys and peaks, however, as reflected in MSE it struggles to predict the highest peaks and lowest valleys. It could be explained polynomial regression not being a good method for this type of data. Reason being the general high variance in the terrain data. The limiting factor observed in the bias-variance trade-off could limit the model from predicting the more extreme values, that is peaks and valleys. Increasing the model complexity seem to result in slightly better performance, but after 20 polynomial degrees the MSE seem to flatten out.

Figure 3 shows the resulting prediction of applying the model to an area of the terrain that the model has not been trained on. It is clear that the prediction is not good. The prediction seem to contain the features of terrain it was trained on. Thus, it is apparent that the trained model is not generalizable to predicting terrain it has not been trained on. We argue that the main reason for this is the complexity of the chosen terrain data. The terrain is not a flat area, it has high peaks and low valleys. Thus requiring a highly complex model, which is not computationally efficient for this assignment to perform. We did increase the number of points and increased the polynomial degree to 35 without seeing any significant improvement in the model. This argument could further supported by the MSE showed in figure 4. MSE being close for both train and test likely could mean that by increasing the complexity to 20 degrees the model fit improves. However, no overfitting is observed, thus indicating that more complexity to result in a better mode. However, we did not find the the most complex model possible because of computational in-efficiency.

B. Ridge

For Ridge regression we plotted MSE for degrees 2 to 19 as a function lambda in figure 8. We can see that MSE is lowest for lambda values of magnitude 10^{-4} (and especially polynomial degree 19), and increase for all polynomial degrees and for increasing lambda. Thus we used this lambda value for the model prediction in figure 7. It seems that the MSE is extremely large, which can be because of the high peaks from the terrain. In the results we also find that the R^2 score go toward the optimal value for both increasing degree and decreasing λ value. But from figure 12 we can see that the most optimal is actually at degree 11. This simply shows that this is the best model-fit for our chosen terrain.

C. Lasso

The Lasso model looks to have a significantly worse prediction than OLS and Ridge. The MSE however is almost the same, so it leads us to believe the predicted elevation are more correct since the shape looks worse. We are not entirely sure why Lasso regression performs worse, and we originally thought that OLS would be the worst of the three methods, however the OLS prediction looks the most accurate and also has the lowest MSE.

D. Bias and Variance Analysis

We were not able to extract the bias variance trade-off for our model using terrain data. The code was slow to run using polynomial degrees above 2. Thus we explored the bias-variance trade-off using the Franke function. This was plotted in figure 13 which was a reasonable reproduction of figure 1 from Hastie et al. (2009). This essentially demonstrates how increased model complexity could result in overfitting. The Franke function was also easier to show overfitting because it is simpler than the terrain data.

E. Cross validation

We can see how MSE changes with respect to increasing polynomial degree for 10 folds used in the resampling method. Generally OLS have the greatest MSE, and Lasso have the best. We see from the figure that MSE increases with model complexity. We also observe that increasing the fold amount k from 5 to 10 decreases the MSE by a magnitude of 10^4. Furthermore, the results indicate that OLS is more sensitive for complexity as we can see in figure 15 and 16 where MSE for OLS shoots up exponentially, while Ridge and Lasso remains relatively low. This could probably be because lambda acts as a shrinkage factor. Thus decreasing the probability of overfitting occuring for both Ridge and Lasso. We used Lasso regression to make a model on the terrain using 10 folds which resulted in 17. Based on visual observation it seemed like the Lasso model was not improved. More speicifically, the Lasso model using 10 folds resulted in MSE =186576, whithout k-fold resampling as shown in figure 10 MSE =186082. Thus, k-fold seem to have caused the model to perform slightly worse, which was not expected. This could be caused by slight variation in code and implementation. But, also the difference

is below 0.2%, thus not significant. Nevertheless, the prediction was not improved as expected.

VI. CONCLUSION

By using the three regression methods OLS, Ridge and Lasso, we tried to find which one could make the best model prediction of a chosen terrain in Norway. We found that OLS ended up being the most accurate prediction with the lowest MSE of 178986.87.

By using OLS regression we found that as we increase model complexity, the MSE of our model reduced by about 13%, while R^2 value went from -6.5 to -5.5, and the β-parameters decreased in absolute value. The optimal polynomial degree which gave the minimum MSE for OLS was 17.

For the Ridge regression we instead found that degree 18 and $\lambda = 10^{-4}$ provided the lowest MSE. The R^2 score is also best for degree 11, given the same λ value.

Using Lasso regression we found that the MSE of our model decreased by about 9%, while R^2 score went from about -6.45 to -5.78. The optimal degree was 17.

Next up we did a bias-variance analysis of the OLS method and demonstrated overfitting by plotting train and test MSE with increasing polynomial degree. We then demonstrated that increasing the degree increased variance and decreased bias.

In the end we used the k-fold cross validation resampling method for the three regression methods. We found that when increasing the numbers of folds k, the MSE of the OLS and Ridge increased more then the MSE of Lasso regression.

From these results we can conclude that OLS seem to be the best model for our specific data set. It had both the best terrain prediction, lowest MSE, most decreasing MSE with respect to increasing polynomial degree, and the greatest change towards optimal R^2 value.

Appendix A. Github repository

https://github.com/LassePladsen/FYS-STK3155-projects/tree/main/project1

Appendix B. List of source code

Here is a list of the code we have developed in this project which can be found in the above Github repository:

- `part_a_franke_OLS.ipynb`
- `part_b_franke_ridge.ipynb`
- `part_c_franke_lasso.ipynb`
- `part_e_bias_variance_OLS.ipynb`
- `part_f_cross_validation.ipynb`
- `part_g_real_data.ipynb`

Appendix C. Analytical derivations

Subsection 1 is for part e of the project, and all the other subsections are for part d.

C1. Bias-variance: equation (9)

This is the derivation of equation (9) for part e of the project. First we write the MSE (7) as the following expectation value

$$MSE = \mathbb{E}[(z - \tilde{z})^2]$$

the we multiply the paranthesis and split the expression to three expectation values. For the rest of these derivations in this project we will write vectors and matrices without boldface for ease of writing $\mathbf{z} = z, \mathbf{X} = X$ etc. We write

$$\begin{aligned} MSE &= \mathbb{E}\left[z^2 - 2z\tilde{z} + \tilde{z}\right] \\ &= \mathbb{E}[z^2] - 2\mathbb{E}[z\tilde{z}] + \mathbb{E}[\tilde{z}^2] \\ &= \mathbb{E}[z^2] - 2\mathbb{E}[z\tilde{z}] + \mathbb{E}[\tilde{z}^2] \end{aligned}$$

We will split this up and take on these three terms one by one, firstly we write $z = f + \epsilon$ which represents our data with noise. For the first term:

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

here we use that $\mathbb{E}[\epsilon] = 0, \mathbb{E}[\epsilon^2] = \sigma^2$ since $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

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

Now for the second term:

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

Now for the third and final term:

$$\mathbb{E}[\tilde{z}^2] = \underline{var[\tilde{z}] + (\mathbb{E}[\tilde{z}])^2}$$

where we have used the definition of variance $var(x) = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$.

We now put these three terms into our expression for

MSE:

$$\begin{aligned}
MSE &= \mathbb{E}\left[(z-\tilde{z})^2\right] \\
&= \mathbb{E}[z^2] - 2\mathbb{E}[z\tilde{z}] + \mathbb{E}[\tilde{z}^2] \\
&= \mathbb{E}[f^2] + \sigma^2 - 2\mathbb{E}[f\tilde{z}] + var[\tilde{z}] + (\mathbb{E}[\tilde{z}])^2 \\
&= \mathbb{E}[f^2] - 2f\mathbb{E}[\tilde{z}] + (\mathbb{E}[\tilde{z}])^2 + \sigma^2 + var[\tilde{z}] \\
&= \mathbb{E}\left[(f-\mathbb{E}[\tilde{z}])^2\right] + var[\tilde{z}] + \sigma^2 \\
&\simeq \mathbb{E}\left[(z-\mathbb{E}[\tilde{z}])^2\right] + var[\tilde{z}] + \sigma^2 \\
&= \underline{Bias[\tilde{z}] + var[\tilde{z}] + \sigma^2}
\end{aligned}$$

here we approximated $f \simeq z$ and used the expressions for Bias (10) and variance (11).

C2. Expectation value of y

We will show that $\mathbb{E}(y_i) = X_{i,*}\beta$ by using $y = f + \epsilon \simeq X\beta + \epsilon$ and separating the expectation value of a sum. Here we approximated f with $X\beta$ using OLS. Then taking a value of y with index i we get $y_i = \sum_j X_{ij}\beta_j + \epsilon_i$:

$$\begin{aligned}
\mathbb{E}(y_i) &= \mathbb{E}(\Sigma_j X_{ij}\beta_j + \epsilon_i) \\
&= \mathbb{E}(\Sigma_j X_{ij}\beta_j) + \mathbb{E}(\epsilon_i) \\
&= \mathbb{E}(\Sigma_j X_{ij}\beta_j) \\
&= \Sigma_j X_{ij}\beta_j \\
&= \underline{X_{i,*}\beta}
\end{aligned}$$

C3. Variance of y

Using the same method as above we will now show $Var(y_i) = \sigma^2$ where σ^2 is the variance of our data's stochastic noise ϵ. Here we use the definition of variance being $Var(x) = \mathbb{E}(x^2) - \mathbb{E}(x)^2$:

$$\begin{aligned}
Var(y_i) &= \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \\
&= \mathbb{E}[(X_{i,*}\beta + \epsilon_i)^2] - (X_{i,*}\beta)^2 \\
&= \mathbb{E}[(X_{i,*}\beta^2 + \epsilon_i^2 + 2X_{i,*}\beta\epsilon_i] - (X_{i,*}\beta)^2 \\
&= \mathbb{E}[(X_{i,*}\beta)^2] + \mathbb{E}[\epsilon_i^2] + \mathbb{E}[2X_{i,*}\beta\epsilon_i] - (X_{i,*}\beta)^2 \\
&= (X_{i,*}\beta)^2 + \mathbb{E}[\epsilon_i^2] + 2X_{i,*}\beta\mathbb{E}[\epsilon_i] - (X_{i,*}\beta)^2 \\
&= \mathbb{E}[\epsilon_i^2] \\
&= Var(\epsilon_i) + \mathbb{E}(\epsilon)^2 \\
&= Var(\epsilon_i) \\
&= \underline{\sigma^2}
\end{aligned}$$

C4. Expectation value of β_{OLS}

Here we will show that the expectation value for the optimal β for OLS, $\hat{\beta}_{OLS}$, equals β_{OLS}:

$$\begin{aligned}
\mathbb{E}(\hat{\beta}_{OLS}) &= \mathbb{E}[(X^TX)^{-1}X^Ty] \\
&= (X^TX)^{-1}X^T\mathbb{E}[y] \\
&= (X^TX)^{-1}X^TX\beta_{OLS} \\
&= \underline{\beta_{OLS}}
\end{aligned}$$

Here we used that the expectation value of the non-stochastic matrix is just the matrix itself ($\mathbb{E}(X) = X$) since it has zero variance (non-stochastic).

C5. Variance of β_{OLS}

Here we will show $Var(\hat{\beta}_{OLS}) = \sigma^2(X^TX)^{-1}$:

$$\begin{aligned}
Var(\hat{\beta}_{OLS}) &= \mathbb{E}[(\hat{\beta}_{OLS})^2] - \mathbb{E}[\hat{\beta}_{OLS}]^2 \\
&= \mathbb{E}[(X^TX)^{-1}X^Ty)^2] - \beta^2 \\
&= \mathbb{E}[(X^TX)^{-1}X^Ty)(X^TX)^{-1}X^Ty)^T] - \beta^T\beta \\
&= \mathbb{E}[(X^TX)^{-1}X^Tyy^TX(X^TX)^{-1}] - \beta^T\beta \\
&= (X^TX)^{-1}X^T\mathbb{E}[yy^T]X(X^TX)^{-1} - \beta^T\beta
\end{aligned}$$

Here we will calculate $\mathbb{E}[yy^T]$ separately for ease:

$$\begin{aligned}
\mathbb{E}[yy^T] &= \mathbb{E}[(X\beta + \epsilon)(X\beta + \epsilon)^T] \\
&= \mathbb{E}[(X\beta\beta^TX^T + \epsilon\epsilon^T + X\beta\epsilon^T + \epsilon\beta^TX^T] \\
&= X\beta\beta^TX^T + \mathbb{E}[\epsilon\epsilon^T] + X\beta\mathbb{E}[\epsilon^T] + \mathbb{E}[\epsilon]\beta^TX^T \\
&= X\beta\beta^TX^T + \mathbb{E}[\epsilon\epsilon^T] \\
&= X\beta\beta^TX^T + \sigma^2I
\end{aligned}$$

Now we put this back into the Variance expression:

$$\begin{aligned}
Var(\hat{\beta}_{OLS}) &= (X^TX)^{-1}X^T(X\beta\beta^TX^T + \sigma^2I)X(X^TX)^{-1} - \beta^T\beta \\
&= [\beta\beta^TX^T + (X^TX)^{-1}X^T\sigma^2]X(X^TX)^{-1} - \beta^T\beta \\
&= [\beta\beta^T + \sigma^2(X^TX)^{-1}] - \beta^T\beta \\
&= \underline{\sigma^2(X^TX)^{-1}}
\end{aligned}$$

Appendix D. Results and discussion for Franke function

Firstly we run the OLS-regression with $N = 1000$ data points. The MSE and R2 score as functions of maximum polynomial degree from $p = 0$ to $p = 5$, in addition to the β_i parameters for each degree p, is plotted in figure 18. Both the MSE and the R2 score decreases with increasing polynomial degree. Comparing $p = 0$ with $p = 5$ the training MSE reduces by about 6% and the test MSE gets reduced by about 5%. The absolute value of train and test R2 scores gets respectively reduced by about 47% and 43%. The beta parameters show how variance and overfitting (greater error) vary with polynomial degree.

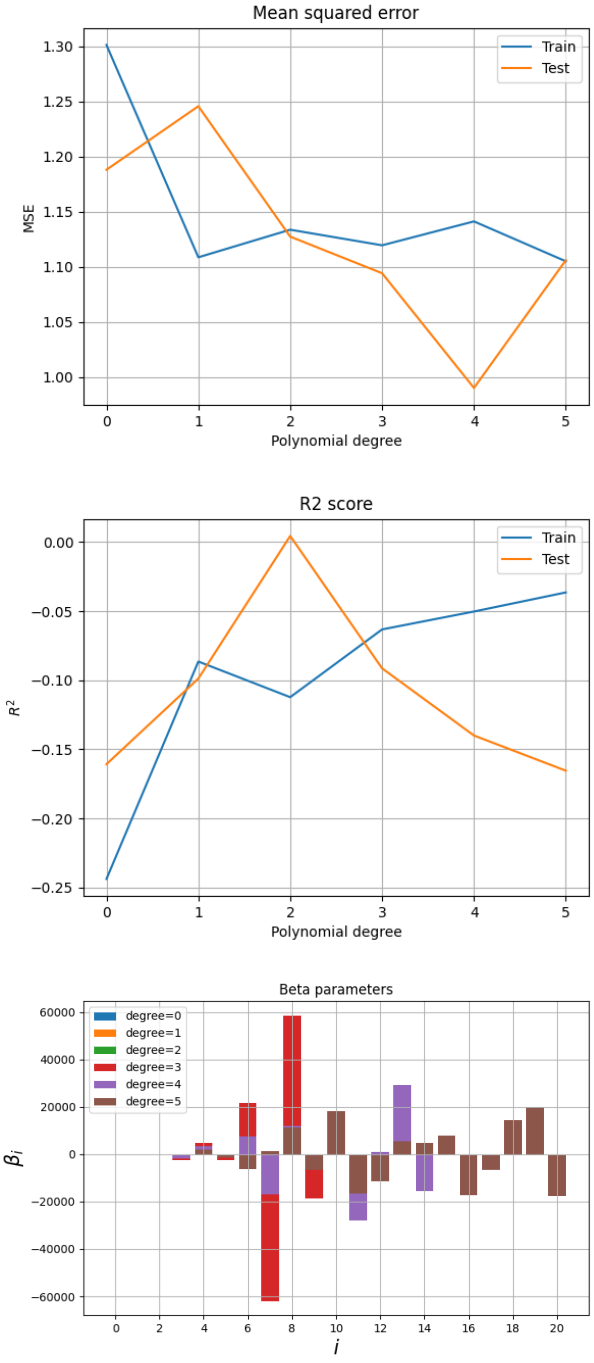

Figure 18: MSE and R2 score for OLS regression on the Franke function with respect to maximum polynomial degree. Also the β_i parameters for each degree is shown as the third plot.

In the plot only the higher complexities have bigger beta values, which tell they have greater variance.

We then repeat this exact same analysis with Ridge regression instead of OLS, and is plotted in figure 19. All MSE gets bigger with incresing λ except for degree four. The same way R2 score goes closer towards the optimal $R^2 = 1$ for decreasing polydegree, but except for degree four.

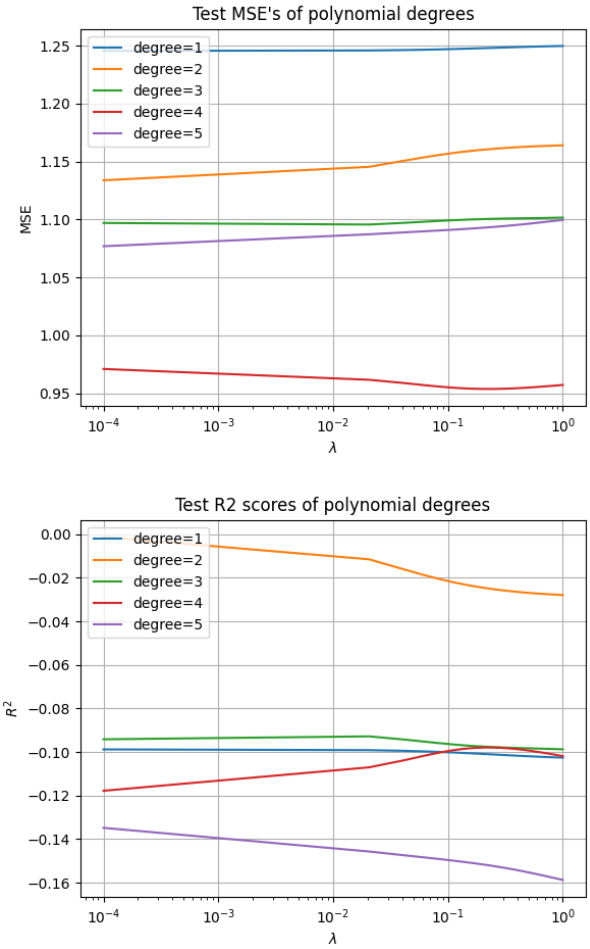

Figure 19: MSE and R2 score for Ridge regression on the Franke function with respect to maximum polynomial degree.

For the Lasso regression we find the following shown in figure 20. Can see big changes up to degree four for both MSE and R2 score. But in total with respect to increasing polynomial degree the training MSE reduces by about 2% and the test MSE gets increased by about 9%. The same way the absolute value of test R2 scores gets reduced by about 15%, while the train R2 score don't change much.

Will now look at analysis of bias-variance-tradeoff for the Franke function with the OLS method in use. We get the results in figure 21. MSE falls off more for the training data over the range of 15 polynomial degrees then

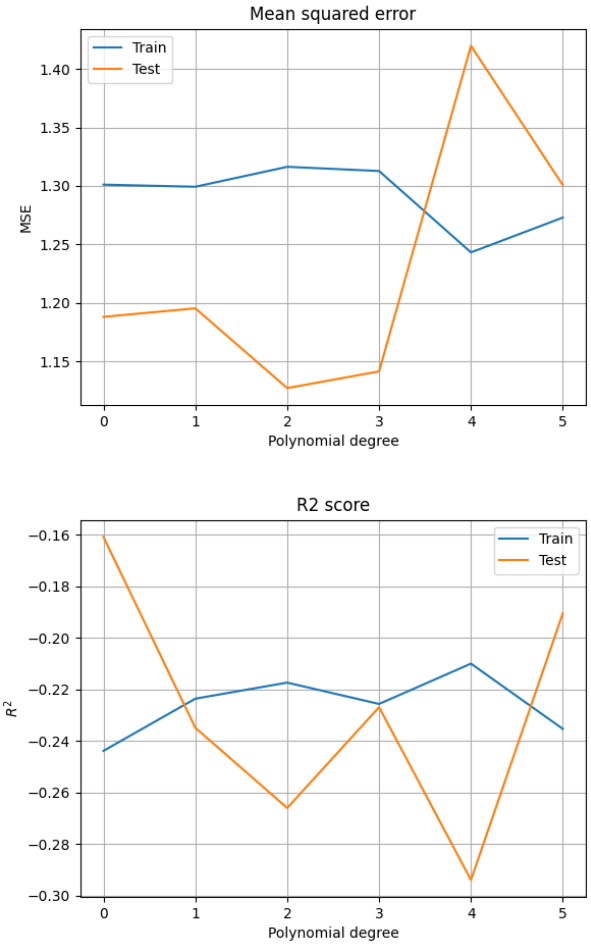

Figure 20: MSE and R2 score of Lasso regression method on the Franke function, with respect to increasing polynomial degree.

the testing data. The result is that the graphs diverge away from each other due to overfitting and we go from high bias and low variance, to low bias and high variance. We then consider degree 11 and varying number of data points, where we generally see a very small bias, which result in MSE and variance more or less overlapping. If we put number of data point set to 800, and instead vary the complexity of the function, we can see how overfitting leads to greater error when the complexity of the function gets to high for our given case.

Thereafter we do the cross-validation, which end up producing the error in figure 22. Here we have done the resampling over $k \in [5, 10]$, but only included $k = 5, 10$ because we see a steady progression of the graphs from the state in $k = 5$ to $k = 10$. Comparing the error of OLS to that when using bootstrap, we can see that for cross validation, MSE increases with respect to greater complexity. Comparing the MSE of the three methods, we see Lasso generally have the least, and that OLS goes from being worst, to catching up to Ridge by increasing number of folds.

References

codebasics. (2019). *Machine learning tutorial python 12 - k fold cross validation [video].* Retrieved 2023-09-29, from https://www.youtube.com/watch?v=gJo0uNL-5Qw

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction.* Springer Verlag, Berlin. Retrieved from https://github.com/CompPhysics/MLErasmus/blob/master/doc/Textbooks/elementsstat.pdf

Hjorth-Jensen, M. (2023). *Applied data analysis and machine learning.* https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/intro.html. ([Online; accessed 30-September-2023])

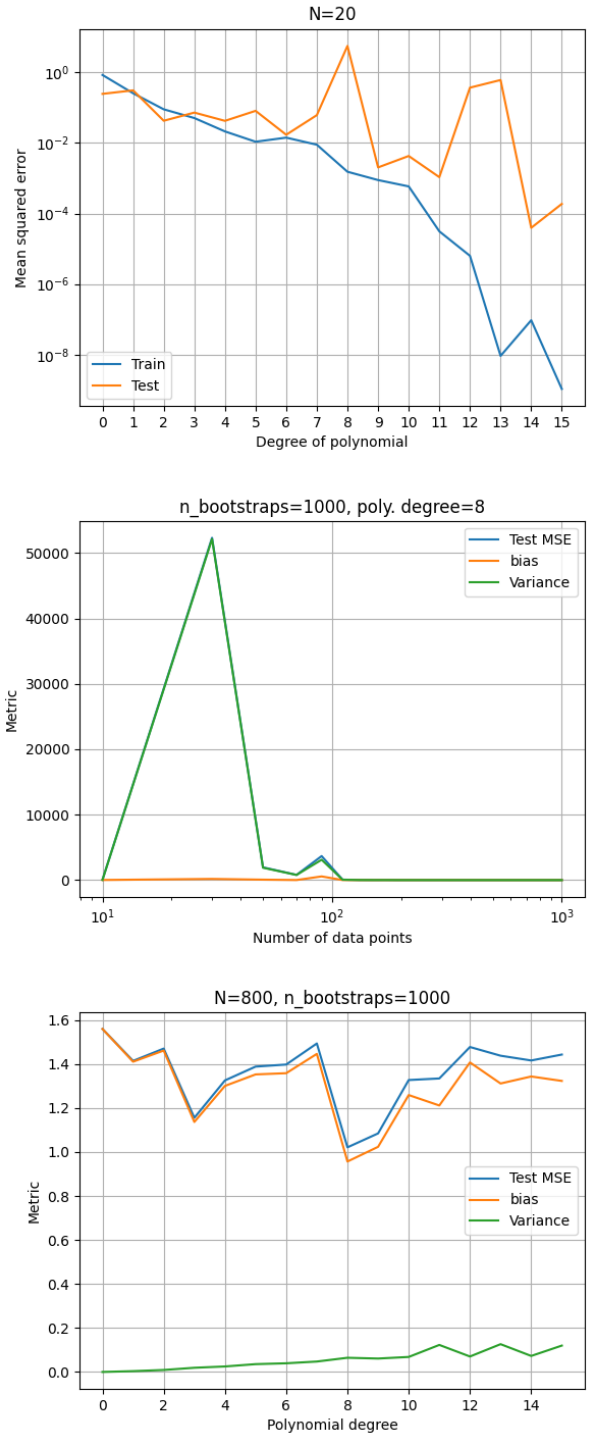

Figure 21: Bias-variance analysis of Franke function using OLS method. From top to bottom; MSE analysis, given polynomial degree analysis, given number of datapoints.

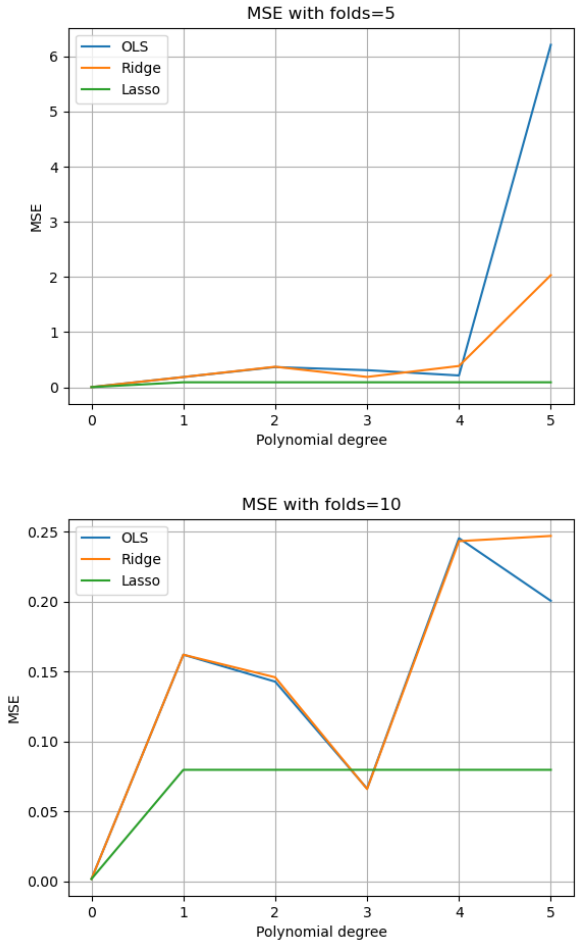

Figure 22: Cross validation of Franke function with $k = 5$ and $k = 10$ using the OLS, Ridge and Lasso methods.