Project 1 on Machine Learning

Oskar Hafstad, Simon H. Hille & Semya A. Tønnessen
(Dated: October 17, 2022)

Github repo for this paper: https://github.com/hafos/FYS-STK4155

The aim of this project is to study linear regression methods by fitting a two dimensional polynomial onto two different datasets. Using Ordinary Least Squares (OLS), Ridge regression and Lasso regression, we attempt to estimate the best fit for the two dimensional function called Franke's function and then for real topographic data. The model is optimized by performing bootstrap resampling and (5-10)-fold cross validation resampling techniques, and we perform an error analysis by studying the Mean Square Error (MSE) and bias-variance trade-off.

For the Franke function we find an optimal model for OLS regression with a polynomial degree $d = 6$. Meanwhile, for the Ridge regression we find both the optimal polynomial, $d = 6$, and saw that the penalty parameter provided a good fit for most values around this area, for a $d \times \lambda$ grid. The MSE score found when using Ridge regression was not notably smaller than the score we found when using the OLS method, and so the OLS regression method is a good fit for the Franke function. The analysis is repeated for the terrain data, taken from a region close to Stavanger in Norway. The best fit using OLS was again found for $d = 6$. For Ridge, the best model was for $d = 8$ for lower values for the penalty parameter. For the Lasso regression we did not obtain proper data, probably due to an error with the scaling which was less visible when studying the OLS and Ridge regression methods.

I. INTRODUCTION

In this report we familiarize ourselves with analysis of linear regression methods, which are methods that attempt to model the relationship between variables an observed data. Linear regression models are often useful, as they are simple and often provide an adequate description of how input affects nonlinear models, particularly when using small numbers of training cases, low signal-to-noise ratio or sparse data.

We will attempt to fit three predictive models, using the methods Ordinary Least Squares (OLS), Ridge regression, and Lasso regression, on two datasets. The first dataset is a grid with values computed using the two-dimensional Franke function, for which we add normally distributed noise $N(0, \sigma^2)$. This function is widely used when testing various interpolation and fitting algorithms. The other dataset consists of terrain based on real topographic data from a region close to Stavanger in Norway. After establishing the model and the method, we employ resampling techniques such as cross-validation and bootstrap, in order to further improve the models and test their validity. We also study the bias-variance trade-off in an attempt to find for which order of polynomials d and a penalty parameter λ we find the best fit for the different models. For Ridge and Lasso regression we expect to see that we can compute more complex models, as these reduce the chance for overfitting. However, these methods are more computationally heavy, and we therefore need to determine the best model depending on the dataset we model.

In section II we provide an overview of OLS, Ridge regression, and Lasso regression, in addition to presenting the datasets and noise, including a description of how we set up the model and design matrix and how we split and scale the data. We also present various methods of evaluating the accuracy of each method. A selection of results relevant to our understanding are presented with a discussion of the results in section III, and in section IV we provide a short summary and outlook.

II. METHOD

We assume the existence of a continuous function $f(\boldsymbol{x})$ and a normal distributed error $\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2\right)$ which describes our data

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ consists of n observed values y_i. We want to use linear regression to approximate the function $f(\boldsymbol{x})$ with a model $\tilde{\boldsymbol{y}}$. There are many different techniques for such calculations, and we will study three of these; one using least squares estimation, and two using penalised estimation.

The function f is approximated by $\tilde{\boldsymbol{y}}$,

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}, \tag{2}$$

where the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the design matrix, containing n rows with vectors $\boldsymbol{x}_i \in \mathbb{R}^p$, and $\beta \in \mathbb{R}^p$ are the unknown parameters we want to determine.

For an observed value y_i,

$$y_i = \sum_{j=0} x_{ij}\beta_j + \varepsilon_i = \boldsymbol{x}_{i,*}\boldsymbol{\beta} + \varepsilon_i, \tag{3}$$

where $*$ means that we iterate over the whole column. The product $\boldsymbol{x}_{i,*}\boldsymbol{\beta}$ is non-stochastic, meaning it is deterministic and therefore $\mathbb{E}[\boldsymbol{x}_{i,*}\boldsymbol{\beta}] = \boldsymbol{x}_{i,*}\boldsymbol{\beta}$. Per definition, $\sigma \sim N(0, \sigma^2)$ is normalized, thereby making $\mathbb{E}[\varepsilon_i] = 0$.

We thus find that the expectation value of $\boldsymbol{y}$ for a given element i is

$$\begin{aligned}\mathbb{E}[y_i] &= \mathbb{E}[\boldsymbol{x}_{i,*}\boldsymbol{\beta} + \mathbb{E}\varepsilon_i] \\ &= \mathbb{E}[\boldsymbol{x}_{i,*}\boldsymbol{\beta}] + \mathbb{E}[\varepsilon_i] \\ &= \boldsymbol{x}_{i,*}\boldsymbol{\beta}. \end{aligned} \tag{4}$$

The variance of this variable is

$$\begin{aligned}\mathrm{Var}[y_i] &= \mathbb{E}[(y_i - \mathbb{E}[y_i])^2] = \mathbb{E}[y_i^2] - \left(\mathbb{E}[y_i^2]\right)^2 \\ &= \mathbb{E}\left[\left(\mathbf{X}_{i,*}\beta + \varepsilon_i\right)^2\right] - \left(\mathbf{X}_{i,*}\boldsymbol{\beta}\right)^2 \\ &= \mathbb{E}\left[\left(\mathbf{X}_{i,*}\boldsymbol{\beta}\right)^2 + 2\varepsilon_i\mathbf{X}_{i,*}\boldsymbol{\beta} + \varepsilon_i^2\right] - \left(\mathbf{X}_{i,*}\boldsymbol{\beta}\right)^2 \\ &= \left(\boldsymbol{X}_{i,*}\boldsymbol{\beta}\right)^2 + 2\mathbb{E}[\varepsilon_i]\mathbf{X}_{i,*}\boldsymbol{\beta} + \mathbb{E}[\varepsilon_i^2] - \left(\mathbf{X}_{i,*}\boldsymbol{\beta}\right)^2 \\ &= \mathbb{E}[\varepsilon_i^2] = \mathrm{Var}(\varepsilon_i) = \sigma^2, \end{aligned} \tag{5}$$

hence, $y_i \sim N(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$, that is $\boldsymbol{y}$ follows a normal distribution with mean value $\mathbf{X}\boldsymbol{\beta}$ and variance σ^2.
We may obtain the optimal estimation, $\hat{\beta}$, of β by minimizing the cost function $C(\boldsymbol{X}, \beta)$. The cost function measures by how much our model deviates from the observed values.

Error Analysis

We calculate two errors, one is the Mean Square Error (MSE), which is the mean value of the square of the errors between approximations and the measured values,

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

as well as evaluating the R^2 score function,

$$R^2\left(\boldsymbol{y}, \tilde{\boldsymbol{y}}\right) = 1 - \frac{\sum_{i=0}^{n-1} y_i - \tilde{y}_i{}^2}{\sum_{i=0}^{n-1} y_i - \bar{y}_i{}^2} \tag{7}$$

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

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i. \tag{8}$$

Bias-variance trade-off

When we have obtained $\hat{\boldsymbol{\beta}}$, we need an estimate of how good our approximation is. For this we use the bias-variance trade-off in the context of continuous predictions such as regression.
In order to find a measure of the expected error in our function we optimize the MSE via the cost function,

$$C(X, \boldsymbol{\beta}) = \frac{1}{n}(y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]. \tag{9}$$

Here the expected value $\mathbb{E}$ is the sample value. One can rewrite the cost function in terms of a term which contains the variance of the model itself. This variance term measures the deviation from the true data and the mean value of the model, called the bias term, and finally the variance of the noise.
We insert $\boldsymbol{y}$ for a model $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}$ into equation 9,

$$\begin{aligned}\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] &= \mathbb{E}[(\boldsymbol{f} + \boldsymbol{\varepsilon} - \tilde{\boldsymbol{y}})^2] \\ &= \mathbb{E}[\boldsymbol{\varepsilon}^2 + 2\boldsymbol{\varepsilon}(\boldsymbol{f} - \tilde{\boldsymbol{y}}) + (\boldsymbol{f} - \tilde{\boldsymbol{y}})^2] \\ &= \boldsymbol{\sigma}^2 + \mathbb{E}[(\boldsymbol{f} - \tilde{\boldsymbol{y}})^2], \end{aligned}$$

where $\boldsymbol{\sigma}^2$ is the error arising due to stochastic noise. We rewrite $\mathbb{E}[(\boldsymbol{f} - \tilde{\boldsymbol{y}})^2]$ by adding and subtracting $\mathbb{E}[\tilde{\boldsymbol{y}}]$,

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

where $\mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{y}}] - \tilde{\boldsymbol{y}}] = 0$ and $\mathbb{E}[(\boldsymbol{f} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] = (\boldsymbol{f} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2$. Inserting into the cost function for the model yields:

$$\begin{aligned}\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] &= (\boldsymbol{f} - \tilde{\boldsymbol{y}})^2 + \mathbb{E}[\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]] + \boldsymbol{\sigma}^2 \\ &= (\mathrm{Bias}[\tilde{y}])^2 + \mathrm{Var}[\tilde{f}] + \sigma^2, \end{aligned} \tag{10}$$

with

$$(\mathrm{Bias}[\tilde{y}])^2 = (\boldsymbol{y} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2, \tag{11}$$

and

$$\mathrm{Var}[\tilde{f}] = \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2. \tag{12}$$

A model having a high bias means that it predicts inaccurate results, even if we only see a small variance in these predictions. Having a lower bias, but with higher variance, implies that out predictions are centered around the true value, but vary markedly. When predicting the model we want to ensure the optimal bias-variance trade-off, in order to get the most trustworthy results.

Ordinary Least Squares

For the OLS method, we assume a cost function

$$C_{\mathrm{OLS}}(\boldsymbol{\beta}) = \frac{1}{n}\left\{(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})\right\}, \tag{13}$$

which, when minimized, yields the OLS expression for the optimal parameter,

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y} = H^{-1}\mathbf{X}^T\boldsymbol{y}, \tag{14}$$

where $H = \mathbf{X}^T\mathbf{X}$ is the Hessian matrix. We can solve this numerically using matrix inversion, if the matrix $\mathbf{X}^T\mathbf{X}$ is invertible.

With the OLS expression for the optimal parameter $\hat{\boldsymbol{\beta}}_{\text{OLS}} = \hat{\boldsymbol{\beta}}$, we can find the expectation value of this parameter,

$$\begin{aligned}\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}] \\ &= \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon})] \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\mathbb{E}[\mathbf{X}\boldsymbol{\beta}]+\mathbb{E}[\boldsymbol{\varepsilon}]) \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta},\end{aligned}$$

where we have that $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}$ is the identity matrix. The variance is

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

where we use that $\mathbb{E}[\mathbf{Y}\mathbf{Y}^\top] = \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^\top\mathbf{X}^\top + \sigma^2\mathbf{I}_{nn}$. We can use this expression to define a confidence interval for the parameters β. A given parameter β_j is given by the diagonal matrix element of the above matrix.

Ridge Regression

The next method we want to analyze is Ridge regression, which is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. The method was developed as a possible solution to the imprecision of least square estimators when linear regression models have highly correlated independent variables. This should provide a more precise ridge parameter estimate, as its variance and mean square estimator are often smaller than other mean square estimators. However, this comes at the cost of computational time.
We define the new cost function to be optimized by adding a penalty term, $\lambda||\beta||_2^2$ to equation 13, where $\lambda \in \mathbb{R}$ is some value such that $\lambda > 0$:

$$C(\mathbf{X},\boldsymbol{\beta})_{\text{Ridge}} = \{(\boldsymbol{y}-\mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\mathbf{X}\boldsymbol{\beta})\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}. \quad (15)$$

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

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{y}, \quad (16)$$

where $\mathbf{I}$ is a $p \times p$ identity matrix where $\sum_{i=0}^{p-1}\beta_i^2 \leq t$, where t is a finite positive number.
We should see that for specific values of λ, we may reduce the variance of the optimal parameters using Ridge regression, which will affect the bias-variance.

Lasso Regression

We once more redefine the cost function, adding a penalty term $\lambda||\boldsymbol{\beta}||_1$ such that we obtain the cost function:

$$C(\mathbf{X},\boldsymbol{\beta})_{\text{Lasso}} = \{(\boldsymbol{y}-\mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\mathbf{X}\boldsymbol{\beta})\} + \lambda||\boldsymbol{\beta}||_1, \quad (17)$$

where $||\boldsymbol{\beta}||_1 = \sum_i|\beta_i|$ and for which we find, by taking the derivative with respect to $\boldsymbol{\beta}$,

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda sgn(\boldsymbol{\beta}) = 2\mathbf{X}^T\boldsymbol{y}. \quad (18)$$

This equation can be solved by using standard convex optimization algorithms [1]. However, performing the calculations is not trivial. We perform the analysis using the functionalities of `scikit-learn` in python.

Resampling

There is often some limit to the amount of data we may obtain, and therefore it will be vital to have methods for resampling this data in order to test it for large samples. Resampling approaches can be computationally expensive, as these involve fitting the same statistical method multiple times using different subsets of the training data. However, due to advances in computing power, this is not always an issue and for certain datasets it may improve the model considerably [1]. We consider two of the most commonly used resampling methods, the bootstrap and cross-validation. These can be used to study the validity of the optimal parameters $\hat{\boldsymbol{\beta}}$, which we obtain using either OLS, Ridge or Lasso regression. The dataset will be divided into train and test data. The training data is used when performing the regression, and the test data is used to evaluate the model.

1. *Bootstrap resampling*

The bootstrap is widely used, as it is generally applicable. It is a non-parametric approach to statistical inference that substitutes computation for more traditional distributional assumptions and asymptotic results. The method is most useful for small datasets, as it generates new datasets and thus increases the amount of data.
We assume that we have some data $\boldsymbol{y}$ from which we estimate $\hat{\boldsymbol{\beta}}$, where $\boldsymbol{\beta}$ has a unknown probability distribution and thus it can be thought of as a random variable where $\boldsymbol{\beta}$ has the highest probability of being $\hat{\boldsymbol{\beta}}$.
We may then draw with replacement n numbers from the data $\boldsymbol{y} = (y_1, y_2, ..., y_n)$. Next we define a vector $\boldsymbol{y^*}$ containing the values which were drawn from $\boldsymbol{y}$, from which we can compute $\hat{\boldsymbol{\beta}}^*$ by evaluating $\hat{\boldsymbol{\beta}}$ under the observations $\boldsymbol{y}^*$. Repeating the process k times yields a list of vectors. The relative frequency of the vectors $\boldsymbol{\beta}^*$ is the estimate of the probability distribution $p(\boldsymbol{\beta})$.
Having obtained k number of parameters for $\boldsymbol{\beta}$, we assume that $\boldsymbol{y}$ is independent and identically distributed

variables so that we may use the central limit theorem, which tells us that the distribution of the parameters should approach a normal distribution for sufficiently large k. The desired test scores can be computed in each bootstrap operation, and in the end the average of these quantities may be used in the model assessment.

Cross validation

Cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance, or to select the appropriate level of flexibility [1].
The bootstrap method has some limitation, as one is not guaranteed to obtain all critical points in the train data. When using cross-validation we ensure that every point makes its way into both the train and test data, and we can sure that we have included everything in the training. This is achieved by randomly re-shuffling the data by splitting the dataset $\boldsymbol{y}$ into k smaller datasets equal in size. One or more of the sets can be used as the testing set, and the remaining will be the training set. We can then fit a model to the training set and evaluate it by computing test scores. This will be repeated k times, or untill all the data has been used as test data.

Datasets

In order to study the different regression methods, we must obtain datasets on which we can perform the analysis. The first dataset we will use is called the Franke function, which is a weighted sum of four exponentials given as,

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &-\frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right), \end{aligned} \tag{19}$$

defined for $x, y \in [0,1]$.
We generate a dataset using a number of uniformly distributed values of $x, y \in [0,1]$, adding some normally distributed noise $\epsilon = N(0,\sigma^2)$. The function which describes our data becomes

$$\boldsymbol{y} = f(x,y) + N(0,\sigma^2). \tag{20}$$

The model for this data, $\tilde{\boldsymbol{y}}$, can be obtained using the method described in section II. The Franke function is visualized without noise in figure 1, and with noise $\sigma = 0.1$ in figure 2. Both figures are plotted on a grid, for uniformly distributed x and y.

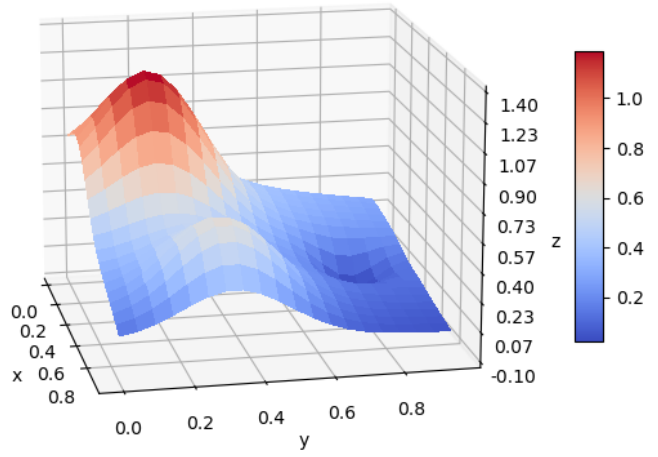

Figure 1. The Franke function for 20×20 uniformly distributed values of $x, y \in [0,1]$.

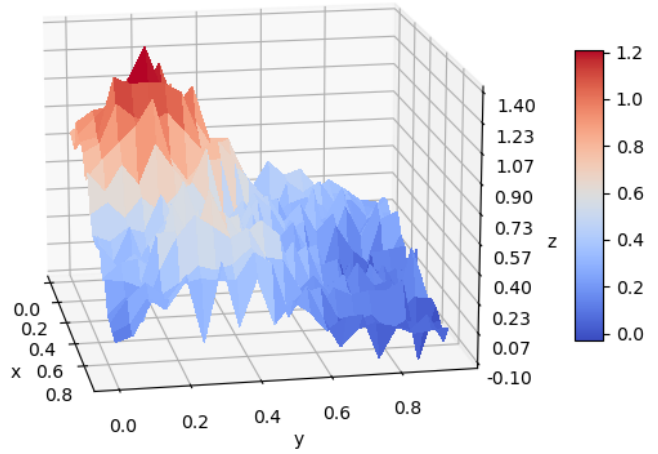

Figure 2. The Franke function for a 20×20 grid of uniformly distributed values of $x, y \in [0,1]$ and noise $\sigma = 0.1$.

We split the dataset at random into two groups, a training set and a test set, using 20% of our dataset for testing and the remaining 80% for training. This is done in order to test the estimate $\hat{\boldsymbol{\beta}}$.
The design matrix, $\mathbf{X}$, has dimensionality $p \times n$, where n is the number of data points and p are the predictors, in our case given by the number of polynomials we wish to include in the fit. The fit has an x and y dependence on the form $[x, y, x^2, y^2, xy, ...]$.
The data obtained may vary somewhat in magnitude. Scaling the data allows us to account for issues with outliers such as particularly large values when we perform our fit, making these points less influential. We scale both the dataset and the matrix. The rescaling method we used is to remove the intercept from the training data.

For a one-dimensional case, the intercept is the value of our output/target variable when all our features are zero and our function crosses the y-axis. For Ridge and Lass removing the intercept influences the MSE greatly because of the regularization time. We observed that this decreases our error.

After testing the methods on this simple function we will perform the same analysis on real topographic data, consisting of digital terrain data downloaded from EarthExplorer [2]. The files are stored in *tif* format which we import into Python using `scipy.misc.imread`.

III. RESULTS AND DISCUSSION

OLS Analysis

We begin by performing a standard OLS regression analysis on the Franke function, using polynomials in x and y up to the fifth order $d = 5$ with a standard deviation of $\sigma = 0.1$. The calculations are done on an $N \times N$ grid. Using equation 14 we are able to compute $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ for polynomials up to the fifth order. These values are plotted in figure 3, with error bars found using the standard deviation found from $\text{Var}[\hat{\boldsymbol{\beta}}_{\text{OLS}}]^{1/2}$ as outlined in II.

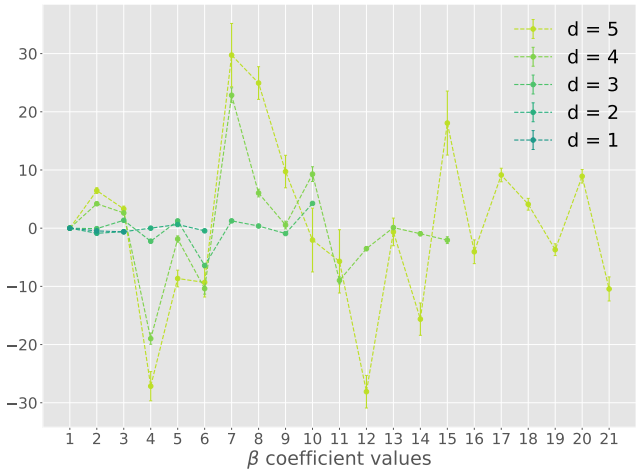

Figure 3. OLS analysis of the Franke function generated with 40×40 datapoints, which shows the values of the β parameters associated with each polynomial of degree d up to $d = 5$. Each point has been plotted with an error margin given by its associated standard deviation σ.

As the complexity of the model increases, it will attempt to cover more points exactly. Therefore, we expect to see the β parameters increase with the polynomial degree of the model. This is in accordance with the figure 3. The model seems to be stable up to $d = 4$, as we also see that the coefficients often have the same sign for the various polynomial degrees.

Having found the optimal parameters $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ we may compute the model $\tilde{\boldsymbol{y}}$ for both the training and test data by using matrix inversion. In order to analyze the error, we compute the MSE and R^2 score using equations 6 and 7, respectively. As we split the data into test and training data, we want to study the MSE arising from both the data reserved for training and the data reserved for testing. The result is presented in figure 4 from which we can see how, as the polynomial degree increases, the MSE score decreases while the R^2 score increases. It is clear that we do not obtain a good fit for the Franke function at lower polynomial degrees. This is as expected, as we are approximating a function with a polynomial, and so the Taylor approximation will become more accurate as the order increases. For this standard OLS analysis, we can also note that the scores obtained from the test and train data do not vary much.

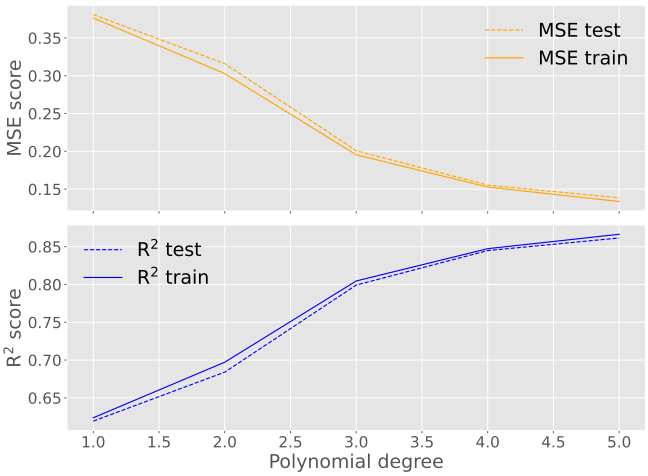

Figure 4. OLS analysis of the Franke function generated with 40×40 datapoints, which shows the MSE and R^2 scores for the train and test data evaluated for polynomials up to order $d = 5$.

To further inspect the accuracy of the model, we want to test it for higher order polynomials. We implement the bootstrap technique with $n = 300$ iterations in order to make the results more reliable, and compute the MSE score for the train and test data, presented in figure 5. From the figure, we see how the train MSE decreases as the model attempts to traverse the data more precisely. However, the test MSE eventually begins to increase rapidly and the model eventually fails for higher polynomials. This is a case of overfitting, where the model is so well fitted to the training data that it becomes unable to make accurate predictions on other data. If we increase the number of datapoints, we will reach the region of overfitting at a larger polynomial degree, due to the greater number of points.

The test MSE can be used to indicate possible regions of low/high bias and variance, and from the graph we judge that the optimal bias-variance trade-off should be around a polynomial degree $d = 6$.

Next, we plot the bias, variance and MSE score for the test data, in order to study the bias-variance trade-off.

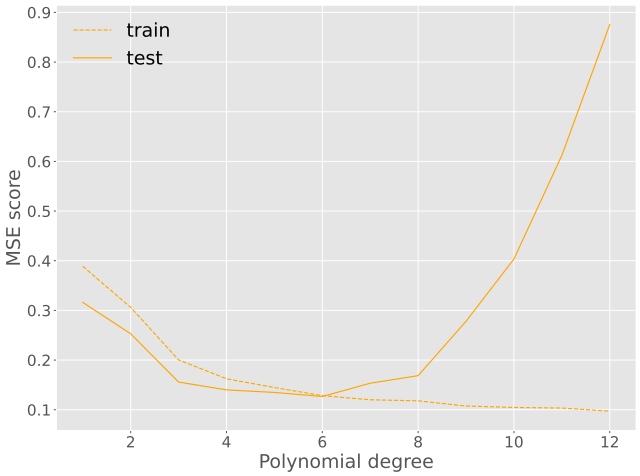

Figure 5. OLS analysis of the Franke function generated with 20×20 datapoints, using a $n = 300$ bootstrap as the resampling technique for polynomial models up to order $d = 12$.

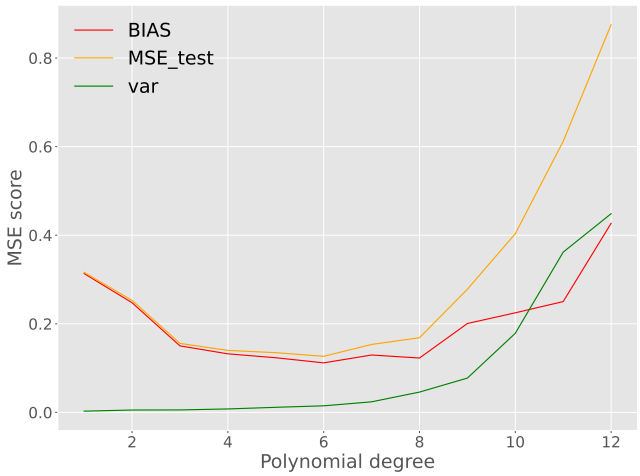

Figure 6. Bias-variance trade off analysis of a 20×20 Franke function where OLS regression has been applied with a bootstrap resampling of $n = 300$. The bias, variance and MSE scores are calculated from the test values of polynomial models up to a degree of $d = 12$.

This allows us to analyse how the error depends on our model complexity in more detail. In figure 6 we see the bias, variance and MSE test error for a bootstrap with $n = 300$ for polynomials up to $d = 12$. For polynomials of lower order we see how the error is due to the bias, while for higher order polynomials the error depends primarily on the variance. It is here we need to make the trade-off, finding the best fit while taking both the bias and the variance into consideration. In this case, the optimal polynomial degree is for a polynomial degree around $d = 6$, as this is where the error becomes dependent on the variance rather than the bias. For the degrees of polynomials we present the bias appears to increase with the variance, but when we investigate higher polynomial models we see that they diverge and that the bias decreases around a model polynomial of degree 18, displaying behaviour as expected for higher model complexity.

Next, we apply the cross-validation resampling technique, using $k \in [5, 10]$-folds of the training data in order to plot the MSE score for both the train and test data. The results are presented in figure 7, from which we see similar features to the bootstrap. The MSE for the training data decreases as expected, while the MSE for the test data reaches a minimum before it begins to diverge. The minimum continues to be around $d = 6$.

A noticeable difference between the bootstrap method and cross-validation is how the MSE for the test values is somewhat lower for the cross-validation method, as can be seen when comparing the figures 5 and 7. The plot when using the cross-validation method is also somewhat smoother. When using the cross-validation method we obtain different train and test data for each fold, as we select random entries for this data. This in turn means that we have more datapoints, thus obtaining a better MSE.

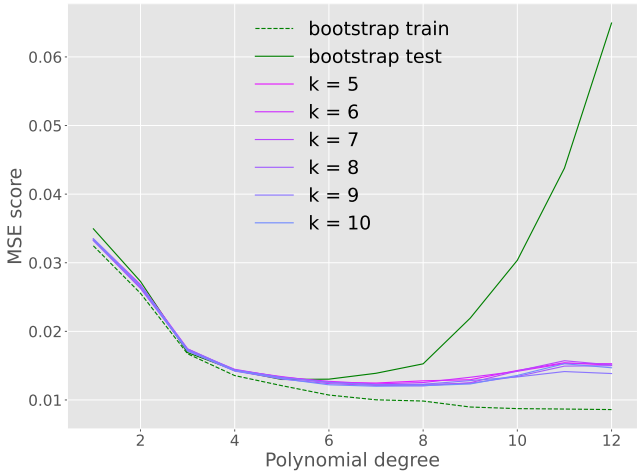

Figure 7. OLS analysis of the Franke function using the cross-validation resampling method with $k \in [5, 10]$ folds.

The optimal polynomial degree for the OLS analysis of the Franke function seems to consistently be at $d = 6$.

Ridge Analysis

We continue our analysis using the Ridge regression method, for which we now have two free parameters, the polynomial degree d and the penalty parameter λ.

In order to study how the MSE varies for both parameters we plot a figure showing how the MSE for the test values varies as a function of $d \in [1, 12]$ and $\lambda \in [10^{-10}, 10^{-1}]$, one using the bootstrap method for $n = 100$ seen in figure 8 and another using cross-fold validation for $k = 10$

seen in figue 9.

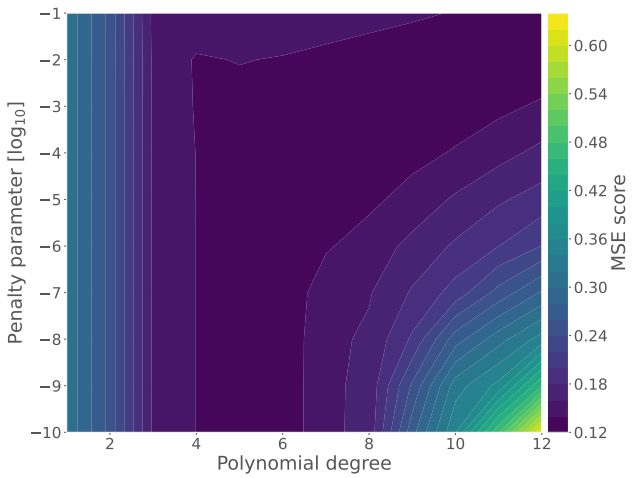

Figure 8. Ridge analysis of the Franke function using a $n = 100$ bootstrap as the resampling technique. The heat map shows the parameter space within which the model has a best fit according to the MSE score obtained.

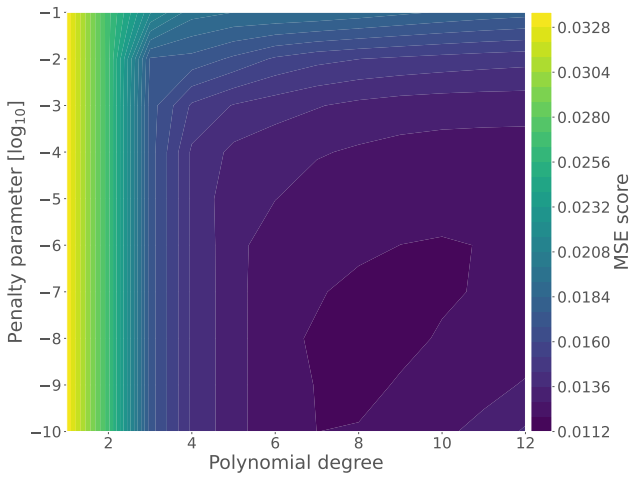

Figure 9. Ridge analysis of the Franke function using cross-validation with $k = 10$ folds as the resampling technique. The plot shows the MSE as a function of the polynomial degree d and the penalty parameter λ.

When the penalty parameter is $\lambda = 0$ we have the regular OLS method, and so we expect to see similar results to OLS for lower values of λ. This seems to be the case for both the Ridge analysis using bootstrap and cross-validation, where the MSE score is at its lowest around $d = 6$ for the bootstrap and around $d = 8$ for the cross-validation, just as we found for the OLS analysis of the MSE score presented in figures 5 for the bootstrap and 7 for the cross-validation. The MSE also begins to increase again for higher polynomials, in accordance with the results for the OLS analysis indicating over or underfitting.

For higher values of λ we see how the penalty parameter restricts which order of polynomial it is that dominates the accuracy of the model. From figure 8 we see how the MSE for polynomials between $d = 1$ and $d = 4$ is greater than for higher polynomial orders. For orders over $d = 6$ we see indications of overfitting, particularely for lower values of λ. For a smaller λ we see an increase in the MSE for higher polynomal degrees, but this is not present as λ increases. This indicates that we only see the bias-variance trade-off for a vary small λ. However, if we increase the maximum polynomial degree, we would see that the MSE begins to increase for higher values of λ.

When performing the same analysis using cross-validation for $k = 10$ folds, the result of which can be seen in figure 9, we get an improvement on what we got with the bootstrap resampling. The MSE is a little higher for lower orders of polynomials, but the model is generally more accurate for higher order polynomials than resampling with bootstrap, an improvement in the MSE on an order of magnitude in fact.

The best fit using cross-validation is found around $d = 8$ and $\lambda \approx 10^{-8}$.

Next, we study the bias-variance trade-off where we vary values of the parameter $\lambda \in [10^{-10}, 10^{-1}]$, as seen in figure. 10.

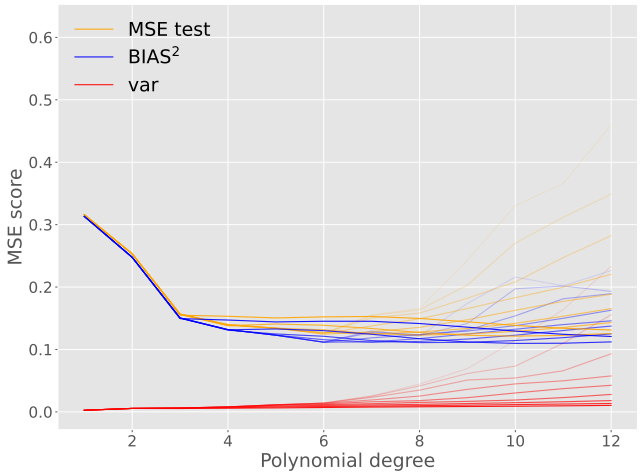

Figure 10. BVT analysis using Ridge regression on the Franke function showing the bias, variance and MSE

Lasso Analysis

Same analysis for Lasso BUT we perform a grid search on a grid of the MSE values, which are functions of both lambda and d in order to find the optimal λ and d at the same time. Next we can plot MSE for the test data predictions.

We see how there is no point in using the Ridge or Lasso methods, as these do not decrease the MSE significantly.

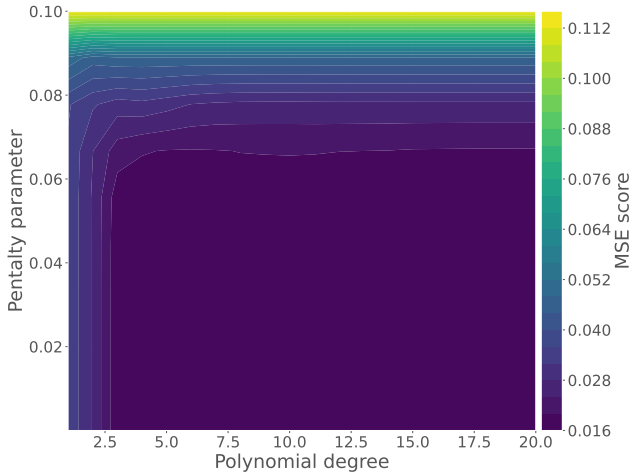

Figure 11. Lasso analysis of the Franke function ($N = 20, \sigma = 0.1$) using a $n = 100$ bootstrap as a resampling technique. The plot shows the MSE as a function of the polynomial degree d and the penalty parameter λ.

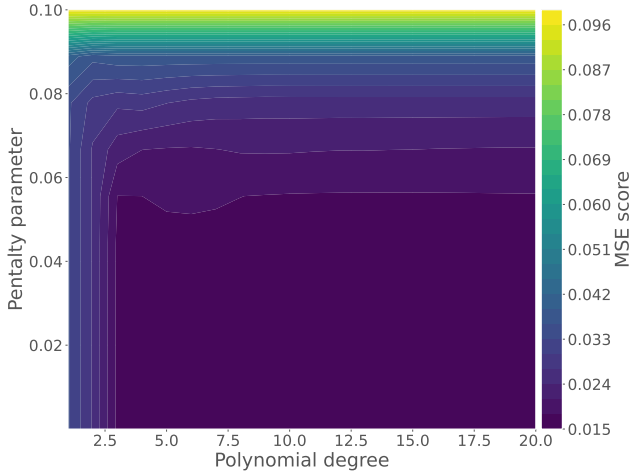

Figure 12. Lasso analysis of the Franke function ($N = 20, \sigma = 0.1$) using cross-validation with $k = 5 - 10$ folds as a resampling technique. The plot shows the MSE as a function of the polynomial degree d and the penalty parameter λ.

The best method is therefore the simple, faster model.

Analysis of real terrain data

After completing our study of the Franke function, we may now perform the same analyses on the terrain data, consisting of a 30×30 grid taken from a region close to Stavanger in Norway. Due to computational limitations, we are unable to model the whole map, and therefore we choose to only model a part of it, as well as performing a data reduction of a factor 10. A scaled image of the terrain data representing the part of the map we model is presented in figure 13.

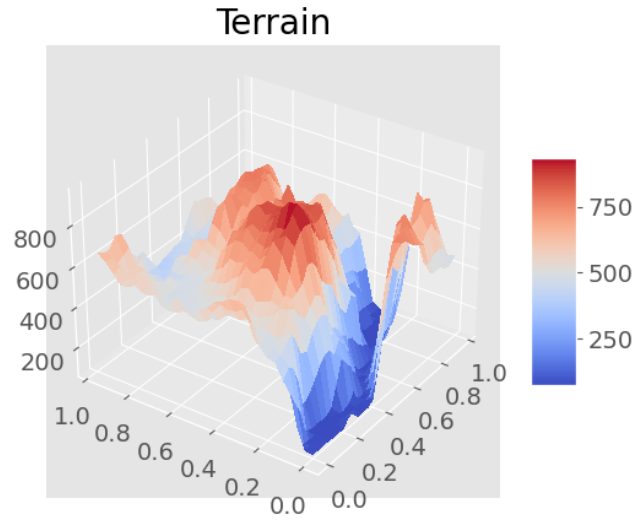

Figure 13. Bias-variance trade off analysis of a 30×30 terrain where OLS regression has been applied with a bootstrap resampling of $n = 300$. The bias, variance and MSE scores are calculated from the test values of polynomial models up to a degree of $d = 12$.

OLS

We begin by performing the same general OLS analysis as we did for the Franke function, plotting the mean of the parameters β for each polynomial degree, presented for $d \in [0, 5]$ in figure 14.

Next, we perform an OLS analysis of the bias-variance for a $n = 300$ bootstrap, figure 15, and of the MSE score for the cross-validation, figure 16.

Again, we see that the model seems to prefer a polynomial degree of $d = 6$, as overfitting begins to set in after this.

1. Ridge

For the Ridge analysis we perform a grid search. As the polynomial degree increases, and the penalty parameter decreases, the calculations will become more computationally expensive, and therefore we try to limit the parameters somewhat.

From figure 17 we see how the model seems best around a polynomial degree of $d = 8$ and for a penalty parameter in the lower part of the map.

Due to time constraints we did not have time to study the Lasso regression method for the terrain data, however this would have been flawed as we know from our study

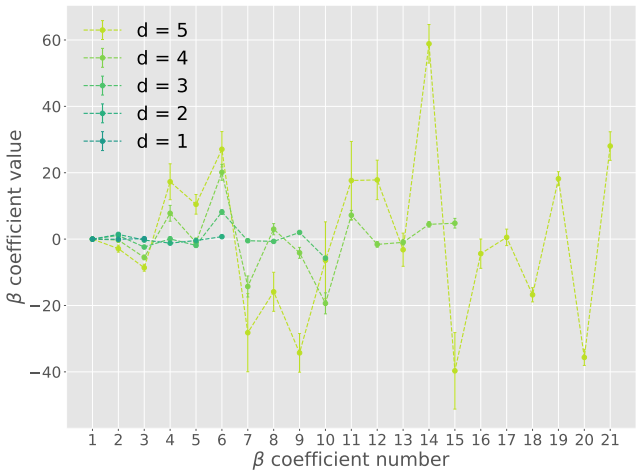

Figure 14. Bias-variance trade off analysis of a 30×30 terrain where OLS regression has been applied with a bootstrap resampling of $n = 300$. The bias, variance and MSE scores are calculated from the test values of polynomial models up to a degree of $d = 12$.

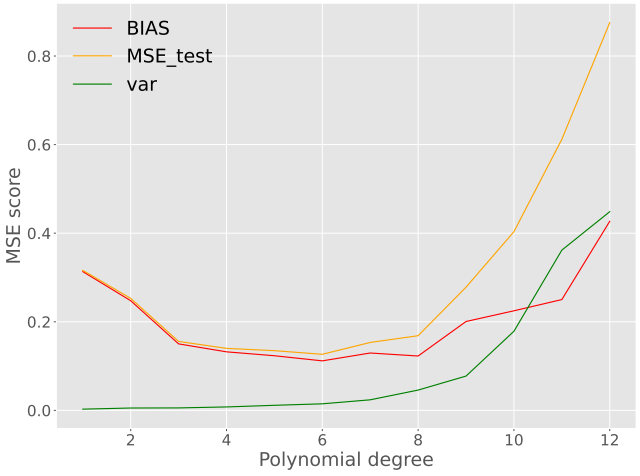

Figure 15. Bias-variance trade off analysis of a 30×30 terrain where OLS regression has been applied with a bootstrap resampling of $n = 300$. The bias, variance and MSE scores are calculated from the test values of polynomial models up to a degree of $d = 12$.

of the Franke function that there is some error in the computation.

Studying the cross-validation might have provided us with a more thorough analysis, but the MSE values we obtained were not sensible, and so we suspect there may be some error in this part of the code. perhaps due to the scaling, which we previously had issues implementing. The terrain data may have points which vary more than the data obtained from the Franke function, and therefore we saw the importance of scaling more clearly when analyzing the terrain data.

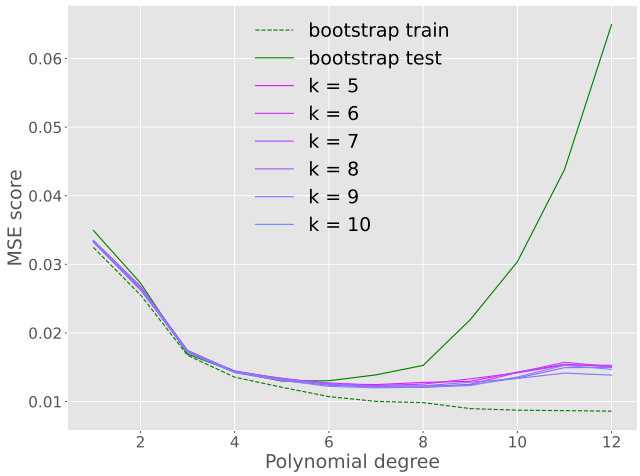

Figure 16. OLS analysis of the terrain using the cross-validation resampling method with $k \in [5, 10]$ folds.

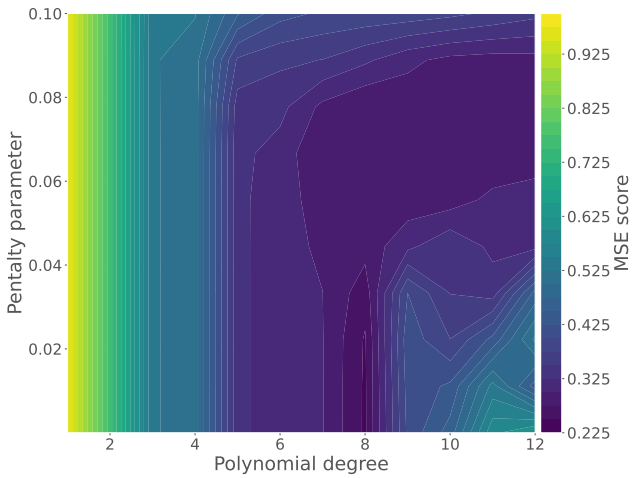

Figure 17. Ridge analysis of the terrain using a $n = 100$ bootstrap as the resampling technique. The heat map shows the parameter space within which the model has a best fit according to the MSE score obtained.

IV. CONCLUSION

In this paper we have studied analysis of three linear regression methods, on two datasets. First, for the Franke function, we found that the OLS method with $d = 6$ was the best fit, as neither Ridge nor Lasso regression improved the model significantly. As the OLS method is the most efficient method, this is a better method for the Franke function. For the terrain data, which consists of a more complex dataset, the Ridge method with $d = 8$ and lower values for the penalty parameter provided us with the best fit.

Cross-validation might have provided a better fit, but for the terrain data the scaling caused this to be inaccurate and so we decided to focus on Ridge regression. Further

studies might try to implement the scaling better, and to study the cross-validation in more detail.

REFERENCES

[1] M. Hjorth-Jensen. *Applied Data Analysis and Machine Learning*. 2021.

[2] US Gov. *EarthExplorer*. Last accessed 16 October 2022. 2022. URL: https://earthexplorer.usgs.gov/.