Department of Physics

FYS-STK4155 - Applied Data Analysis and Machine Learning

Project 1

A linear regression model for terrain data

Authors:
Stig Patey and Jouval Somer

October, 2022

Table of Contents

List of Figures ii

List of Tables iii

1 Introduction 2

2 Theory 3

2.1 OLS Approximation . . . 3

2.2 RIDGE Regression . . . 6

2.3 LASSO Regression . . . 6

2.4 Resampling with Bootstrap . . . 7

2.5 Resampling with K-Fold Cross Validation . . . 7

2.6 Model Selection and Model Assessment . . . 7

2.7 Approximating a smooth function with a polynomial . . . 8

3 Method 8

3.1 Data Sets and Overview of Methods . . . 8

3.2 2D Polynomial Regression . . . 8

3.3 Scaling . . . 9

3.4 OLS . . . 10

3.5 Ridge . . . 10

3.6 LASSO . . . 10

3.7 Process Overview . . . 11

3.8 Validation setup for testing the methods on the Franke Function . . . 11

3.9 Data and Data set, Spliting and Resampling setups . . . 12

3.10 The hyperparameter λ and polynomial degree range . . . 12

3.11 Results for the Franke function . . . 13

3.11.1 OLS . . . 13

3.11.2 RIDGE . . . 15

3.11.3 LASSO . . . 17

3.12 Comments to the Results from the Franke Data . . . 18

3.13 Setup for the Real Terrain Data . . . 19

3.13.1 Larger Data Set . . . 19

3.13.2 LASSO Parameters . . . 19

3.13.3 Increase in Maximum Polynomial Degrees . . . 19

4 Results **20**

4.1 OLS . . . 20

4.2 RIDGE . . . 22

4.3 LASSO . . . 24

4.4 Plots . . . 25

4.4.1 Optimum Model Against Polynomial Degree and λ . . . 25

4.4.2 Lowest MSE Achieved . . . 25

5 Discussion **26**

5.1 Bias-Variance Tradeoff . . . 26

5.2 Predictor Coefficients (betas) . . . 26

5.3 Heatmaps and Model Selection . . . 27

5.4 Scaling Issues . . . 27

5.5 Computational restrictions . . . 27

6 Conclusion **28**

Bibliography **29**

Appendix **30**

A Github Repository . . . 30

List of Figures

1 Number of beta coeff. per polynomial degree. . . . 9

2 Surface plots of FF without and with noise. . . . 11

3 MSE against polynomial degree for OLS. No resampling used. . . . 13

4 MSE against polynomial degree for OLS. Resampling = cross validation . . . 13

5 Bias–variance tradeoff for OLS. Resampling = bootstrap. . . . 14

6 R2 score against polynomial degree for OLS. No resampling used. . . . 14

7 Plot of β_0, β_1 and β_2 as a function of polynomial degree. No resampling used. . . . 15

8 Bias–variance tradeoff for RIDGE. Resampling = bootstrap. . . . 15

9 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap . . . 16

10 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation . . . 16

11 Bias–variance tradeoff for LASSO. Resampling = bootstrap. . . . 17

12 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap 17

13 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation 18

14 MSE against polynomial degree for OLS. No resampling used. 20

15 MSE against polynomial degree for OLS. Resampling = cross validation 20

16 R2 score against polynomial degree for OLS. No resampling used. 21

17 Bias–variance tradeoff for OLS. Resampling = bootstrap. 21

18 Plot of β_0, β_1 and β_2 as a function of polynomial degree. No resampling used. . . . 22

19 Bias–variance tradeoff for RIDGE. Resampling = bootstrap. 22

20 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap 23

21 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation 23

22 Bias–variance tradeoff for LASSO. Resampling = bootstrap. 24

23 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap 24

24 Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation 25

25 Ratio of beta coeff versus data points per polynomial degree. 26

List of Tables

1 Optimum model against associated polynomial degree and λ for each regression model and resampling method. 25

2 Lowest MSE against associated polynomial degree and λ for each regression model and resampling method. 25

Abstract

In this project we apply regression methods such as OLS, Ridge and LASSO to gain insight in a data set of terrain data. We use two-variable polynomials to fit the data at various order, and tune the hyperparameter λ for RIDGE and LASSO. We then perform resampling with both non-parametric bootstrap and k-fold cross validation. In this process we split the data in test and train sets and perform model assessment on the train data, with the aim to find the best model to predict the unseen data in the test set. We here explore the bias-variance tradeoff. All the regression methods are first validated using the MSE and R2 score and applied on a known smooth function, the Franke function with Gaussian noise, before the same methods are used and re-tund on the more irregular terrain data. We finally conclude that Ridge regression provides the best fit to the terrain data.

1 Introduction

The main aim of this project is to study various regression and resampling methods. Analyzing data is not a new venture, humans have for thousands of years explored their surroundings and tried to gain insight through observations and experiments. Ancient astronomers and explorers took measurements and made data records. It could be the time a comet passed, the water level at tide or the periodic observations of sunspots that Galilei observed. By recording their measurements, they created what we call data sets. At some point, having enough data, it could become evident that there was some sort of patterns in the data. Perhaps there was a linear or quadratic relationship, perhaps a trigonometric function could describe the tide? Perhaps the sun rotated at a given angular velocity? Extracting useful knowledge out of data, and eventually be able to make predictions required some sort of geometric intuition at first and eventually more advanced mathematical methods were developed.

Having collected some data, how do we proceed? One intuitive thought is that a good model, in the sense that it describes the data well, should fit the data closely. A fitted model line should be near the measured data points. Here we need to find a suitable measure that describes quantitatively what we mean by near. Intuitively this could be the distance (error) from a data point to the fitted model line, and to ensure certain nice properties, such as differentiability, the square of this distance is commonly used as a metric. The mean sum of squares of errors (MSE) is then a measure, and this is the essence of the very first regression method we will deploy, the OLS (Ordinary Least Squares) method.

Legendre first published the method of least squares in 6th March 1805 in a work on determining the orbits of comets. Gauss published his version of the least squares method in 1809 and, although acknowledging that it had already appeared in Legendre's book, Gauss referred to the method of least squares as "our principle" (principium nostrum in Latin) and by claiming that he, Gauss, had been using the method since I795. This led to one of the infamous priority disputes in the history of mathematics.(Plackett 1972)

Within ten years after Legendre's publication, OLS had been adopted as a standard tool in astronomy and geodesy in France, Italy, and Prussia, and by 1825 in England. The rapid geographic diffusion of the method is a success story that has few parallels in the history of scientific method (Plackett 1972).

With OLS we are able to explore correlations between variables. An important note here for all regression methods is that correlation does not mean that we can conclude that there is any clear cause-effect relationship between the variables. Any correlation we find could be attributed to other variables we have not included in our analysis, or the cause-effect relationship could go in the reverse direction from what we assume. For an educational and amusing example in this regard the reader is referred to New evidence for the Theory of the Stork (Höfer et al. 2004).

Data sets can be low or high dimensional. Traditionally we may have data sets where the number of observations (n) are larger than the number of predictive variables (p). But for high dimensional data sets, we may have a small set of observations, for example a medical study with 100 test patients, and millions of possible predictive parameters based on their DNA. We could also have a data set where many variables are strongly correlated. Such data sets may require newer methods such as Ridge and LASSO.

The model we find is an approximation of the phenomena we investigate. The purpose of regression is thus not the to make an exact model, that is probably impossible, but rather to make a useful model, a model that can provide insight on the correlations between the explanatory variables and the output. To quote the British statistician George E.P. Box: "All models are wrong but some are useful" (Box 1979).

We will apply these three linear regression methods, OLS, RIDGE and LASSO, on two data sets. One test set, acquired by Franke's function, is simply to validate our methods and models, before we analyse a set of digital terrain data. For each method we will create several models, and then assess their performances. Finally based on this assessment we perform model selection, and select the ideal model for modelling the terrain data.

2 Theory

2.1 OLS Approximation

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

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

We then approximate this function with our model from the solution of the linear regression equations, ordinary least squares (OLS), that is our function f is approximated by $\tilde{\boldsymbol{y}}$ where we minimized $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$, the mean square error (MSE), with

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

The matrix $\boldsymbol{X}$ is the so-called design or feature matrix.

We assume that $\boldsymbol{X}$ and $\boldsymbol{\beta}$ are deterministic, which means that $\tilde{\boldsymbol{y}}$ is deterministic.

$\boldsymbol{X}$ has a row for each observation, consisting of 1 and then the values of the $p-1$ predictors. The equations relating the observed y_i's to the x_i's can then be written as:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1p-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{np-1} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

We have now found an $n \times m$ matrix X and the vector $\boldsymbol{y} \in \mathbb{R}^n$. Our approximation problem can then be expressed as:

$$\text{find } \boldsymbol{\beta} \in \mathbb{R}^p \text{ such that } \|X\boldsymbol{\beta} - \boldsymbol{y}\|_2 \text{ is minimal.}$$

This minimization is obviously the same as minimizing the square of the norm. By squaring the norm we can use rules of linear algebra and differentiation in the following way:

$$\|X\beta - y\|_2^2 = (X\beta - y)^\top (X\beta - y) = \beta^\top X^\top X\beta - 2\beta^\top X^\top y + y^\top y$$

Now for notation let us denote the left hand side as the loss function $L(\beta)$ and differentiate L w.r.t. β:

$$L(\beta) = \|X\beta - y\|_2^2$$

$$\frac{\partial L(\beta)}{\partial \beta} = \frac{\partial \left(\beta^\top X^\top X\beta - 2\beta^\top X^\top y + y^\top y\right)}{\partial \beta} = -2X^\top y + 2X^\top X\beta$$

We have found an expression for the derivative of the loss function, we now set this to zero and solve for $\boldsymbol{\beta}$:

$$X^\top X\boldsymbol{\beta} = X^\top \boldsymbol{y}$$
$$\boldsymbol{\beta} = \left(X^\top X\right)^{-1} X^\top \boldsymbol{y}$$

This is called the OLS normal equations (Lay et al. 2016). The design matrix X is typically ill-conditioned, and the effects of rounding errors in a numerical solution by matrix inversion can then result in a solution that is far from exact (Süli and Mayers 2003). Therefore we typically use SVD or QR algorithms, which will be described in the **Method** section.

We will now consider a given element i in (1): $y_i = f(x_i) + \varepsilon_i$

The expectation value of $\boldsymbol{y}$ for a given element i

$$\mathbb{E}\left(y_i\right) = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\boldsymbol{\beta},$$

and its variance is

$$\mathrm{Var}\left(y_i\right) = \sigma^2$$

Hence, $y_i \sim N\left(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2\right)$, that is $\boldsymbol{y}$ follows a normal distribution with mean value $\boldsymbol{X\beta}$ and variance σ^2.

With the OLS expressions for the parameters $\boldsymbol{\beta}$ we will now show that

$$\mathbb{E}(\boldsymbol{\beta}) = \boldsymbol{\beta}.$$

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= E\left[\left[X^\top X\right]^{-1} X^\top Y\right] \\
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \left[X^\top X\right]^{-1} X^\top E[Y] \\
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \left[X^\top X\right]^{-1} X^\top E[X\beta + \varepsilon] \\
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \left[XX^\top X\right]^{-1} X^\top X E[B] + 0 \\
\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \beta
\end{aligned}$$

This means that the OLS estimator of the regression parameters is unbiased.

We will finally show that the variance of $\boldsymbol{\beta}$ is

$$\mathrm{Var}(\boldsymbol{\beta}) = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}.$$

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

We will now show that we can rewrite this in terms of a term which contains the variance of the model itself (the so-called variance term), a term which measures the deviation from the true data and the mean value of the model (the bias term) and finally the variance of the noise. By rules calculating with expectations, we get:

$$\begin{aligned}
\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \mathbb{E}\left[(f(\boldsymbol{x})+\boldsymbol{\varepsilon}-\tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}}+\boldsymbol{\varepsilon})^2\right] \\
&= \mathbb{E}\left[(\tilde{y}-f(x))^2+2(f(x)-\tilde{y})\varepsilon+\varepsilon^2\right] \\
&= \mathbb{E}[\tilde{y}-f(x)]^2+2\mathbb{E}\left[f(x)-\tilde{y}\right]\varepsilon+\mathbb{E}\left[\varepsilon^2\right] \\
&\text{the mid term is 0, and the last term is } \sigma^2\text{, so we simply get:} \\
&= \mathbb{E}[\tilde{y}-f(x)]^2+\sigma^2
\end{aligned}$$

Now we can add and subtract $\mathbb{E}[\tilde{\boldsymbol{y}}]$ from the first term in order to split this term in two separate parts:

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

Again the mid term is 0, so we get, by calculation rules of expectations:

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

Hence we have shown that:

$$\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] = (\text{Bias}[\tilde{y}])^2+\text{var}[\tilde{f}]+\sigma^2,$$

with

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

and

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

2.2 RIDGE Regression

With a large number of features, OLS may have a low bias but high variance when compared with models with a smaller number of features.

In such cases we could theoretically make a perfect fit on the trained data set using OLS, but the model would most likely be useless as a prediction on unseen data. We would have an overfitted model, and even if we selected less parameters, to avoid overfitting, it is possible that keeping more parameters but applying some sort of regularization on them would provide a better model overall. One method that explores this concept is ridge regression, developed around 1970 (Hoerl and Kennard 1970).

By shrinkage towards the mean we can limit the effect of certain parameters in the model, adjusting their importance though a hyperparameter we call λ. The effect or ridge shrinkage is that we do not exclude any parameters in ridge but their effect is dampened and approach zero as λ goes to infinity.

The cost function for Ridge is (Azzalini and Scarpa 2012):

$$L(\beta, \lambda) = \|X\beta - y\|^2 + \lambda\beta^\top\beta$$

We see that the last term provides an L2-norm. Again we differentiate L w.r.t. β by similar calculations as shown in the OLS section and get:

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

If $\lambda = 0$ then Ridge is equivalent to OLS, as can be seen by the equation above.

As noted in the section on Scaling, we see that the scale of the features in the design matrix will impact the ridge regression. For this reason scaling is required.

2.3 LASSO Regression

An effect of ridge shrinkage is that we do not exclude any parameters. This is a clear disadvantage with ridge regression, if the data set has a large amount of predictors. First of all, any model should be useful for the end users. Having an extremely large volume of predictors may make the model hard to understand and maintain, also considering the amount of data we must collect to provide the required input to the model and the computational power needed. If many of these predictors have a very small value to the model, a better alternative could be to simply remove some of the parameters and provide a simpler model.

This takes us to the next method we will explore, the LASSO (least absolute shrinkage and selection operator). It is a more recent method, first introduced in geophysics (Santosa and Symes 1986) and then popularized as a method in 1996 by Tibshirani (R. Tibshirani 1996). Most notably the method uses the L1 norm instead of the L2 norm, and by forcing the sum of the absolute values of the regression coefficients below a certain threshold we end up excluding some of them, thus performing selection of parameters.

It can be shown that the L1 norm is the lowest that still yields a convex problem. Convexity simplifies the computation as does a model with less predictors. (Hastie et al. 2015).

The cost function for LASSO is (Azzalini and Scarpa 2012):

$$L(\beta, \lambda) = \|X\beta - y\|^2 + \lambda \sum_{j=1}^{p-1} |\beta_j|$$

There is no analytic solution and the minimization of the loss function must be solved numerically by iterative methods. We will use SciKit Learn LASSO for this. This algorithm uses coordinate descent to optimize (SciKit Learn LASSO).

Again, as noted in the section on Scaling, we see that the scale of the features in the design matrix will impact the LASSO regression. For this reason scaling is required.

It should be noted that we cannot a priori know whether Ridge or Lasso performs the best. If many predictors are assumed to be irrelevant then LASSO may be the better option, while in data sets with predictors that all have relevance then Ridge may perform better (James et al. 2013)

2.4 Resampling with Bootstrap

It would be great to have an abundance of data at hand, but that is not always the case. An experiment could be expensive to conduct, or we may have limited observations of a physical event. In such cases we can use resampling techniques to extract as much information from the real data as possible.

Any resampling technique is of course reliant on the quality of the real data. If we have poor quality in our real data to start with, due to measurements errors or outliers, then no resampling can rectify that.

One resampling technique is bootstrap (Efron and R. J. Tibshirani 1994) where we randomly draw datasets with replacement from the training data, each sample the same size as the original training set. This is done B times, producing B bootstrap training dataset. Then we fit our model to each such data set.

Very seldom are more than $B = 200$ replications needed for estimating a standard error. The actual number of replications needed will also depend on the size of the data set. Larger values of B are also required for bootstrap confidence intervals (Efron and R. J. Tibshirani 1994).

2.5 Resampling with K-Fold Cross Validation

Another resampling technique we will employ is cross-validation. K-fold Cross validation splits the data set in K non-overlapping sets that we call folds. Each fold will in succession act as the test set, and the union of the remaining $K - 1$ sets will act as training sets. We shuffle the data then create the folds.

2.6 Model Selection and Model Assessment

By model selection we find the model with the optimum flexibility and by model assessment we evaluate the performance of our model(James et al. 2013).

Model Selection involves selecting the *best* model. *Best* could have different meanings, for our purpose in this report we aim to identify the model with the lowest test MSE after resampling. This relates to the bias-variance trade off, described in the Theory part as the lowest test MSE is found in the region where we have neither under fitting or over fitting.

In many practical cases it should be noted however that if the test MSE curve is relatively flat, a somewhat simpler model should be favoured for simplicity. A selection of an MSE within one

standard error of the lower MSE value is typically a practitioners method. The reason is that for example with a different K-fold split we could get slightly different result (James et al. 2013).

2.7 Approximating a smooth function with a polynomial

We will generate data points from the Franke function adding gaussian noise.

The Franke function itself is continuous, differentiable and real valued. From real analysis in mathematics we note that we can approximate such a smooth function arbitrarily close by a polynomial. We could use Taylor expansions and more generally by the Stone–Weierstrass theorem we can get arbitrarily close to any continuous function with a polynomial. (Lindstrøm 2017).

Consequently, since the underlying function is a smooth function, it can be expected that we are able to fit a polynomial well to the Franke data points, even with some noise added.

Conversely, we cannot expect the same benefit when exploring the highly irregular terrain data. With vertical cliffs there may not exist any continuous function for the terrain.

3 Method

3.1 Data Sets and Overview of Methods

In this project we have two data sets that we will approximate with a 2D polynomial using OLS, Ridge and LASSO regression. The first data set is self-generated data from the Franke Function, a known smooth function used here to validate our methods, before we continue to the second data set, terrain data from a place in Norway, recorded by NASA's Shuttle Radar Topography Mission (SRTM).

For the second data set we downsample the data for computational reasons, the full data set is too large for our available computer resources. This will be detailed below.

Finally we will perform resampling on these methods with Bootstrap and Cross Validation. The entire project is coded in Python and made available on this projects (Github Repository).

3.2 2D Polynomial Regression

We assume that the response z in the Franke Function, and later the z (height) in the terrain data can be approximated by a 2D polynomial $z = f(x, y) + \epsilon$, where f takes the (x, y) location on a rectangular surface (X,Y) as input. Note that we do not split the (X,Y) surface in parts with separate functions, hence our function f(x,y) will act globally on this space.

For a n-th order polynomial, counting all the interaction variables, the number of terms in the polynomial increase quickly. The number of β's as a function of the polynomial degree can be calculated as $(n^2 + 3n + 2)/2$.

This can be seen in the following plot, and this gives an indication of the complexity of the model as the polynomial degree is increased.

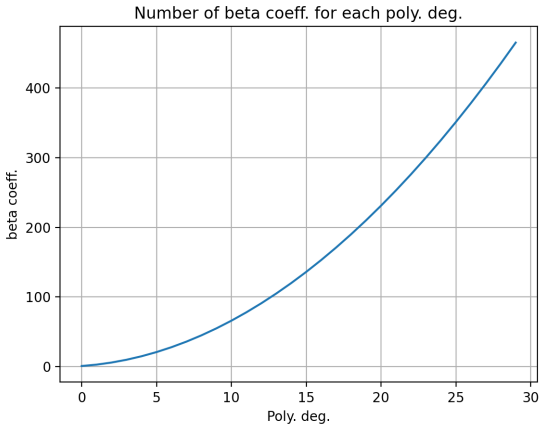


Figure 1: Number of beta coeff. per polynomial degree.

3.3 Scaling

The features in the design matrix can be measured in units that differ greatly in their numerical range.

For OLS the least squares coefficient estimates are scale equivariant: multiplying Xj by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$ (James et al. 2013). For this reason scaling of the design matrix is not needed in OLS mathematically. However, numerically there would be concerns related to the limitations of the float numbers available on a computer. Assume for example a model that takes as Xj the distance from the earth to the sun in meters. This is roughly the Astronomical Unit (AU), $AU \approx 1.5e + 11$. Now in a 30th degree polynomial we could then get an overflow, since the largest float value is typically $1.8e+308$, which we found by running a small test script. In such case we would need to scale Xj even before it is raised to the nth power for the design matrix.

For our Franke Function and for our Terrain data we decided to scale the features even for OLS. We noted that the code performed better when scaled.

For scaling we will use the SciKit Learn StandardScaler(). We have also tested this scaler by developing our own code for scaling and compared the results which were equivalent. This code is available on our Github repository.

For Ridge and LASSO Regression scaling of the design matrix is required, since the magnitude of each β_j is affected by the scale (as stated for OLS), but this time our minimization of the squares of the beta-coefficients in ridge regression or our minimization of the absolute value of the beta-coefficients in LASSO regression will be impacted by their magnitude. Hence scaling and centering each predictor is needed.(James et al. 2013).

We could out of convenience also scale the response variable Z. By the linearity of linear regression any scaling of the response will just scale the least squares coefficient estimates by the same scaling. Hence it will not affect the accuracy of our regression, but it could be beneficial for the interpretation of the model and comparison across models on different data sets. For this reason we will scale the response as well.

We tried to implement scaling directly on the design matrix but got unreliable results. We therefore moved scaling to the pre-processing stage, and did not scale the design matrix after the variables had been raised up to the n-th power.

3.4 OLS

As detailed in the Theory Section 2 in the Ordinary Least Squares (OLS) method we create the design matrix X and then solve the minimization problem:

$$\text{find } \boldsymbol{\beta} \in \mathbb{R}^p \text{ such that } \|X\boldsymbol{\beta} - \boldsymbol{y}\|_2 \text{ is minimal.}$$

We will solve this numerically with our Python code. While the normal equations to solve the minimization of OLS are important from a theoretical perspective, X is likely to have a high condition number (be ill-conditioned) and by squaring the entries in X we get that $X^T X$ will be even more ill-conditioned.

To avoid methods that will cause numerical issues we instead use orthogonal methods such as the SVD or QR algorithms which are more stable (Süli and Mayers 2003). For this we implement in our code built in methods in Numpy, such as numpy.linalg.pinv() for SVD and numpy.linalg.qr() for QR factorization, where the QR solution is then found by numpy.linalg.solve(), which is faster than inverting the matrix.

Finally for validation purposes we also tested our own code for QR factorization, where we did not rely on the Gram-Schmidt process, but instead an iterative process based on the induction proof in Theorem 2.12 of (Süli and Mayers 2003). Our own QR factorization code is a bit slower than the Numpy method, but gave the same results.

3.5 Ridge

For the Ridge method we proceed in the same manner as for OLS, but now we test for various λ hyperparameters.

As detailed in Theory Section 2 for ridge we get:

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

As noted in Scaling Section 3.3, we will use scaling for ridge regression.

3.6 LASSO

For the LASSO (Least Absolute Shrinkage and Selection Operator) method we proceed in the same manner as for Ridge, but now we use scikit-learn Lasso to solve the minimization problem given by the cost function for LASSO, Theory Section 2,

$$L(\beta, \lambda) = \|X\beta - y\|^2 + \lambda \sum_{j=1}^{p-1} |\beta_j| \,.$$

We experienced some convergence warnings and problems when running scikit-learns LASSO function for low tolerances. Therefore we set the maximum iterations to 1000 and the tolerance to $0,05$.

As noted in Scaling Section 3.3, we see that the scale of the features in the design matrix will impact the LASSO regression. For this reason scaling is required.

3.7 Process Overview

We proceeded in the following sequence for this report:

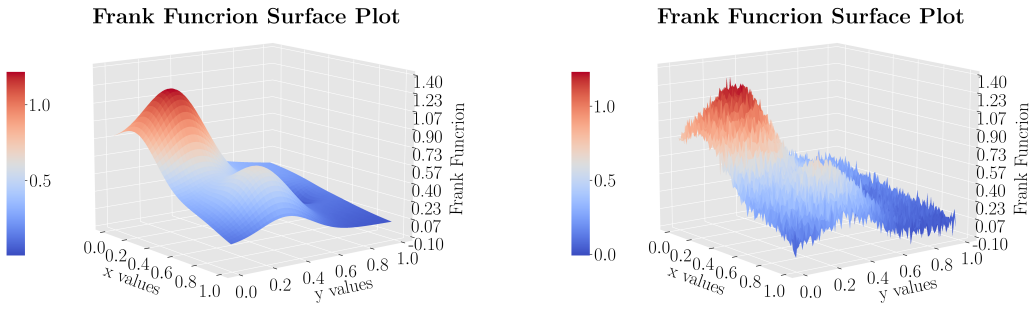
- Validation of the regression methods on the Franke Function
- Validation of resampling methods on the Franke Function
- Perform the same methods on downsized Terrain Data

3.8 Validation setup for testing the methods on the Franke Function

As a proof of concept and means to verify the regression methods used to create a polynomial fit of the terrain data, we tested the three methods first on a known three dimensional exponential function, the Frank Function (FF), see (3).

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

The x-y grid was generated by meshing two numpy arange arrays that span from 0 to 1 with a given step size parameter. Lowering the step size will yield more data points and a larger data set, and vice versa. Creating a larger data set is not normally something that is easy to do with real data, although we will explore re-sampling methods that simulate this, such as bootstrap (BS) and cross validation (CV) later. The mesh was then fed into the FF to create a smooth data set as seen in Figure 2a. Then, normally distributed noise with mean 0 and variance 1 was scaled, with a parameter α, and added to the smooth data to mimic real noisy data, see Figure 2b.



(a) Smooth FF without noise.

(b) Noisy FF with scaling parameter $\alpha = 0.05$.

Figure 2: Surface plots of FF without and with noise.

Now that we had our test data we could start analysing the three regression methods: OLS, RIDGE and LASSO. This was done to make sure our implemented methods worked and behaved according to the analytical and expected results we got in Section 2 (Theory).
We validated our methods primarily using the mean square error (MSE), but also using the coefficient of determination (R^2). The resulting scores (MSE and R^2) were then plotted as functions of the polynomial degree. Additionally, we plotted the coefficients $\boldsymbol{\beta}$ against polynomial degree to

observe the fluctuations in the parameters. Large fluctuation in $\boldsymbol{\beta}$-s could give an indication of overfitting as this implies that the model is trying to trace all the given datapoints for each new polynomial degree added.
We also plotted a bias-variance tradeoff plot to get an overview of how the penalty term in RIDGE and LASSO affect the variance on higher polynomial degrees.
For the regression methods RIDGE and LASSO we plotted "zoomed in" heatmaps of MSE against the regularization parameter λ and polynomial degrees. This was done for the two resampling techniques (BS and CV), around the lowest MSE achieved. Thus, giving an easy to read visual plot of how the MSE behaves around the best λ-s and polynomial degrees. To find the lowest MSE we used numpys argwhere function and their min function. This gave us the coordinates (in the λ/polynomial degree grid) of the lowest MSE. The first-coordinate being the optimal polynomial degree and the second-coordinate being the optimal λ. Then, to acquire a spread of λs and degrees to plot we multiplied the optimal values with 0.5 and 1.5 to get respectively the min and max values of the plot. The floor and ceiling functions from numpy was used on the multiplications for min and max respectively.

3.9 Data and Data set, Spliting and Resampling setups

In order to split our data into train and test sets we utilized the scikit-learn function "train_test_split()". This function was passed with test_size = 0.2 (yielding a 80/20% train test split) and a given seed for easy replicability. Within the bootstrap resampling technique the scikit-learn function "resample" was used to shuffle the train data after already splitting into train and test sets. Thus, leaving a test set completely outside the bootstrap for later validation. One could also do the train-test-split inside each bootstrap replicate, but this was not done in this project. Further analysis is done in the Discussion Section 5. The number of bootstrap recompilings, B, was sat to $B = 50$, and k the number of folds for cross validation was sat to $k = 10$. The selection of $B = 50$ was due to a larger data sets, with a 80/20% train test split of terrain data we still had a large data set in both training and testing, and trials with computational speed and consistency in our results made us arrive at $B = 50$.
Furthermore, the step size on the numpy arange arrays was sat to 0.05 which gives 20 values for x and 20 for y generating 400 data points.

3.10 The hyperparameter λ and polynomial degree range

The regularization parameter λ had a wide range with 18 values, ranging from 10^{-12} to 10^{5}, one for each power of ten.
And, the range of polynomial degree was sat to 15 for OLS and RIDGE and 20 for LASSO for the final runs. Due to LASSOs L_1 penalty terms tendency to squeeze some of the predictor coefficients to zero, and having a "max budget" for their total, we expect the variance to stay small for a larger amount of polynomial degrees thus giving a larger leeway.

3.11 Results for the Franke function

3.11.1 OLS

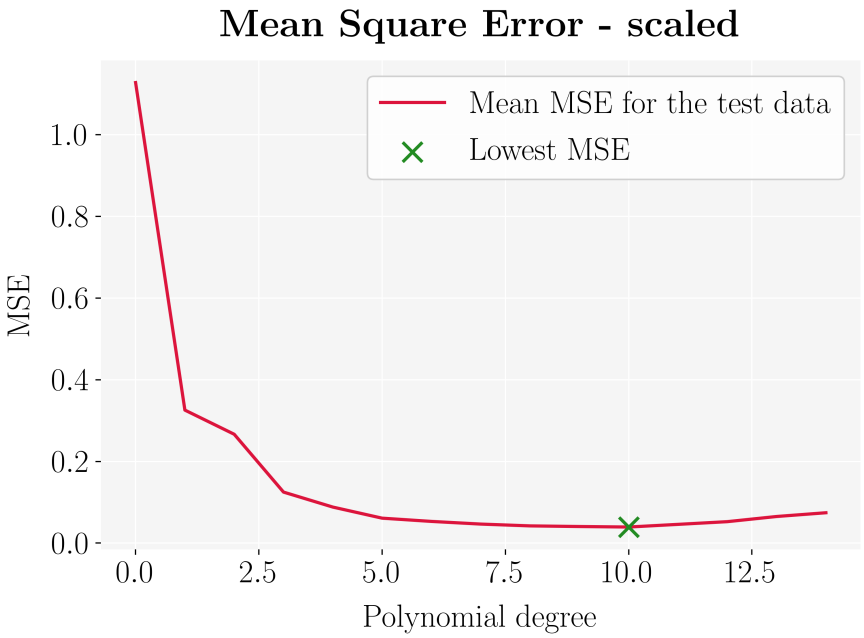


Figure 3: MSE against polynomial degree for OLS. No resampling used.

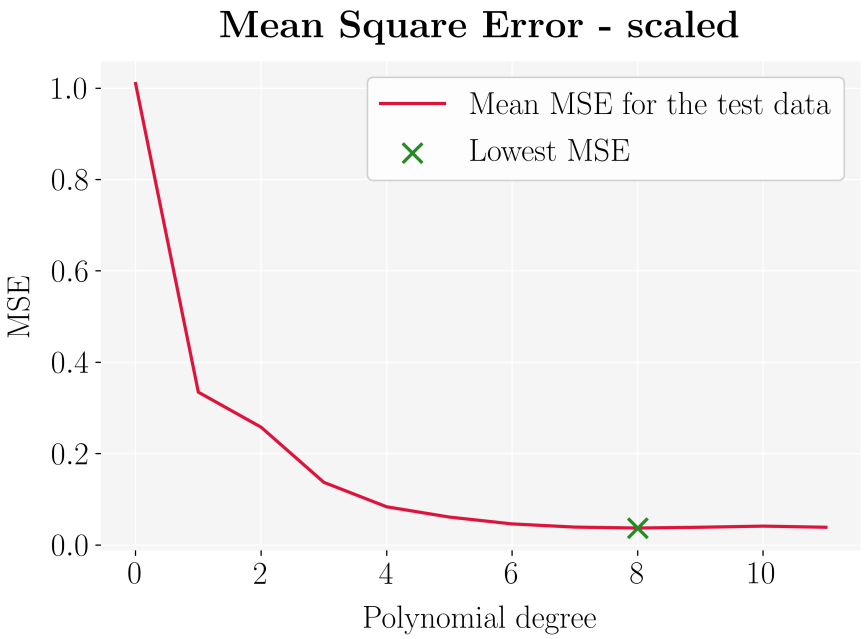


Figure 4: MSE against polynomial degree for OLS. Resampling = cross validation

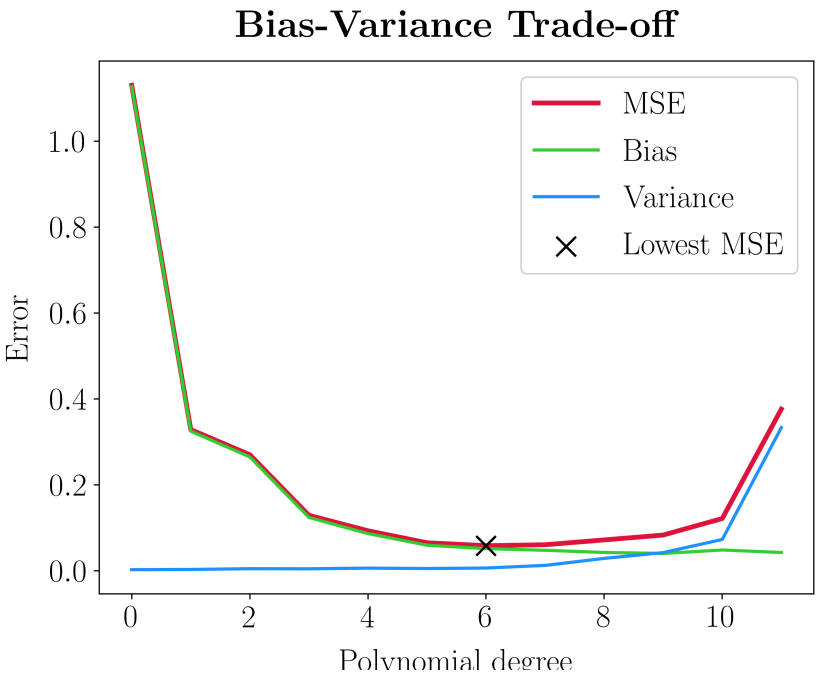


Figure 5: Bias–variance tradeoff for OLS. Resampling = bootstrap.

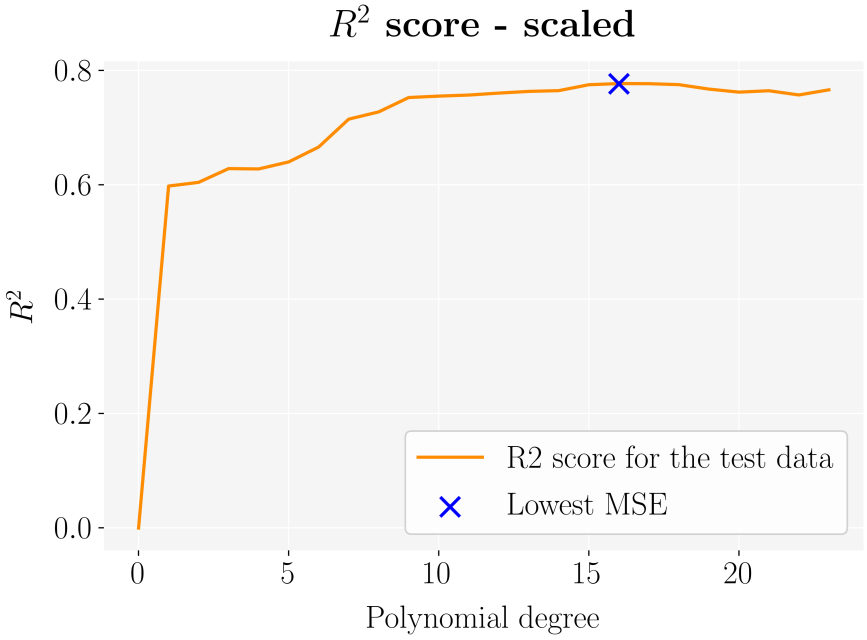


Figure 6: R2 score against polynomial degree for OLS. No resampling used.

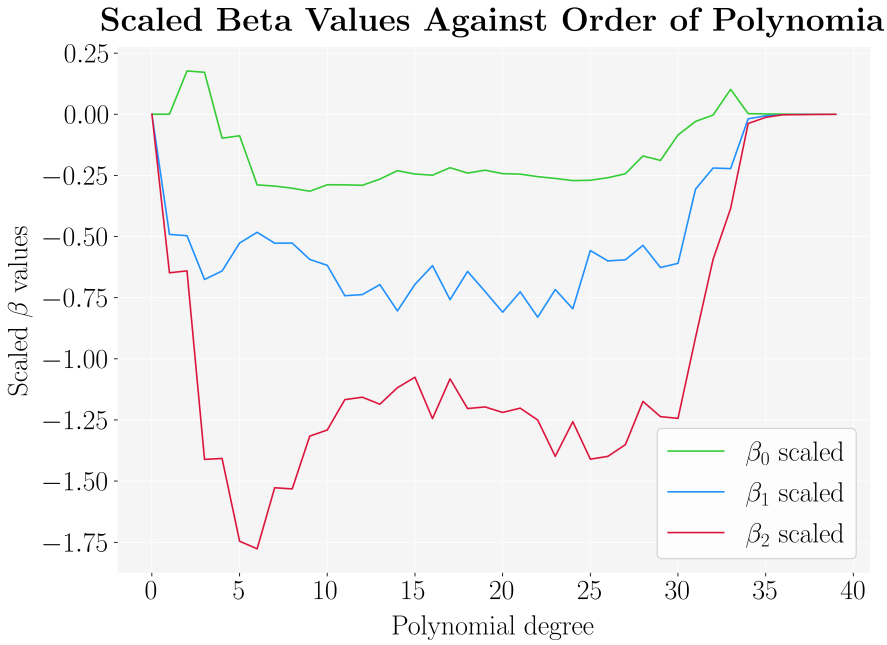


Figure 7: Plot of β_0, β_1 and β_2 as a function of polynomial degree. No resampling used.

3.11.2 RIDGE

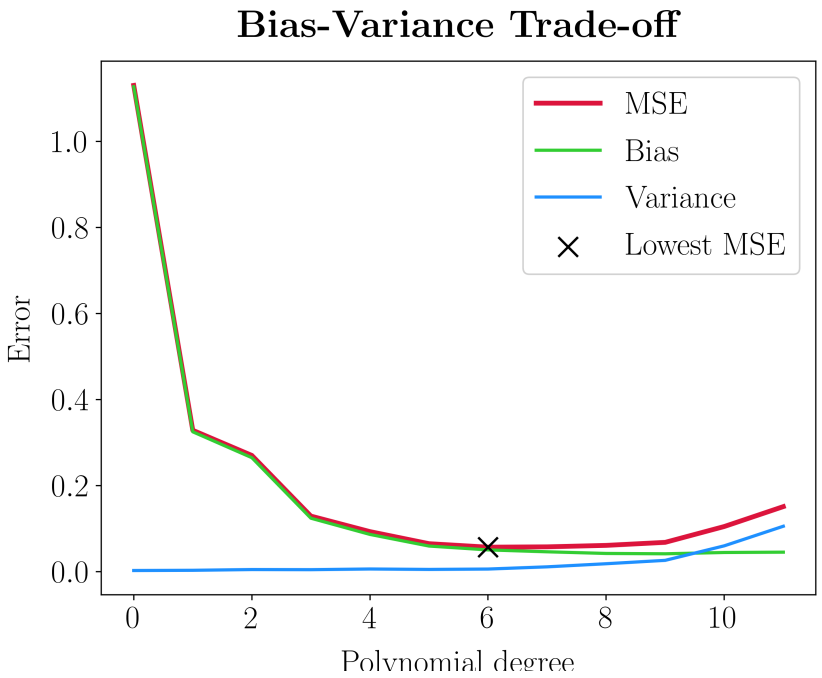


Figure 8: Bias–variance tradeoff for RIDGE. Resampling = bootstrap.

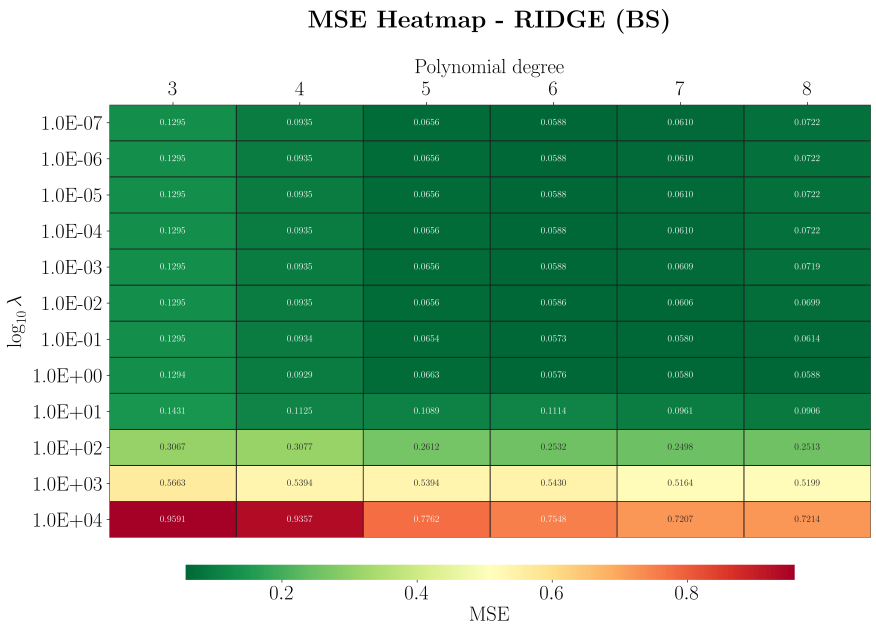


Figure 9: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap

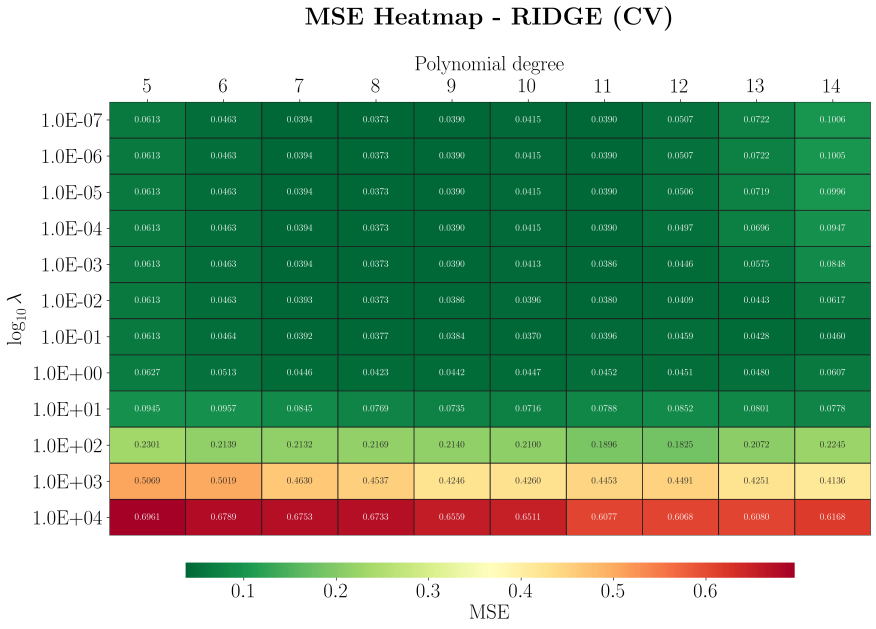


Figure 10: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation

3.11.3 LASSO

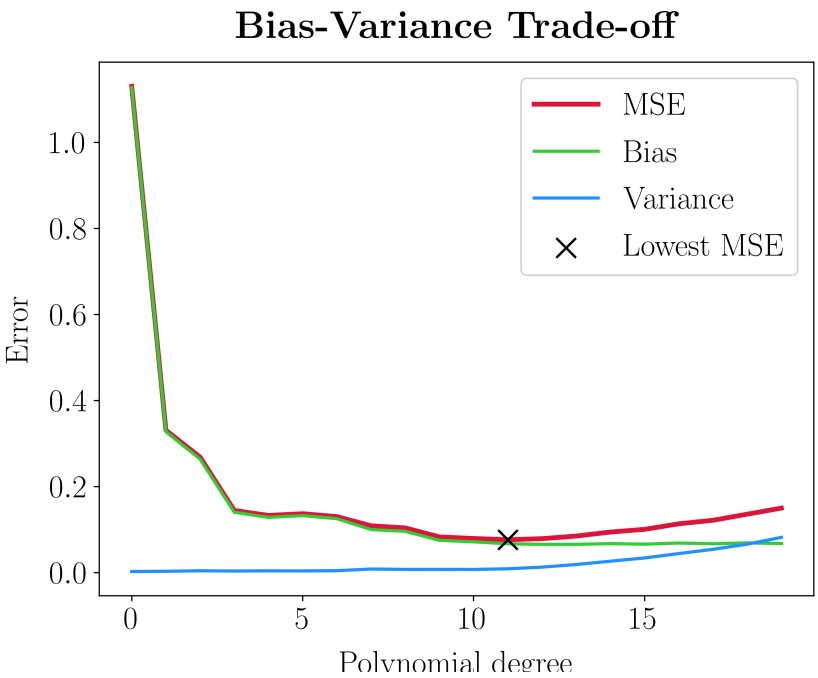


Figure 11: Bias–variance tradeoff for LASSO. Resampling = bootstrap.

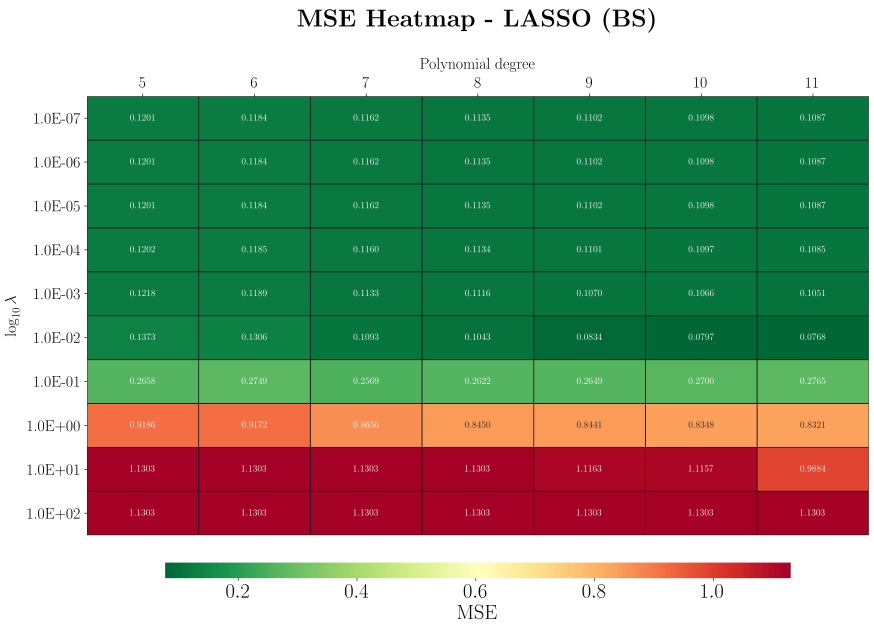


Figure 12: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap

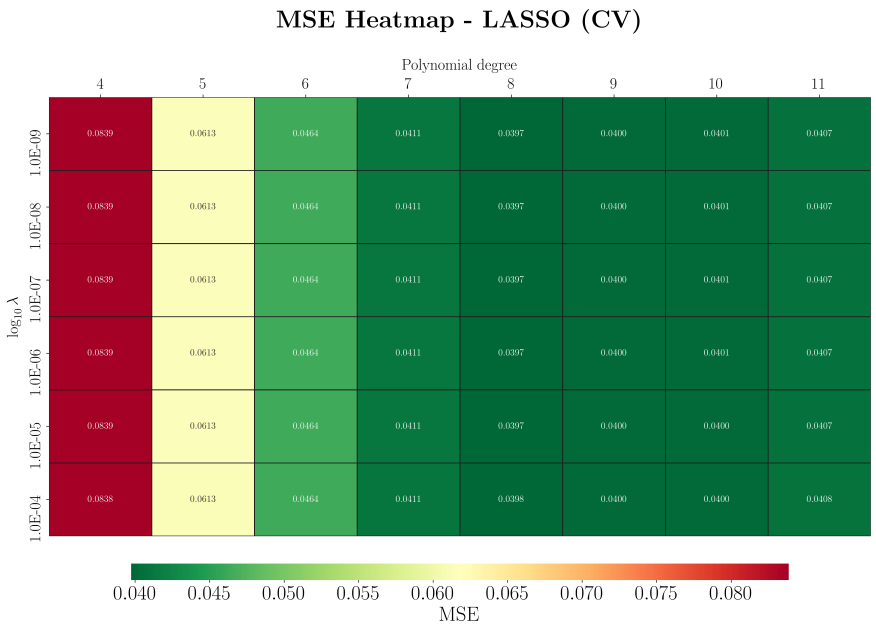


Figure 13: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation

3.12 Comments to the Results from the Franke Data

We have used the Franke data to validate our the methods. We noticed that a bias-variance trade off can be seen at increasing complexities, with a clear overfitting for higher orders. With BS the lowest OLS MSE was 0.0588 at polynomial degree 6. Ridge achieved MSE at 0.0573 with BS at polynomial degree 6 at $\lambda = 0.1$. With our tuning parameters set LASSO achieved MSE 0.0768 using BS at polynomial degree 11 = 11 at $\lambda = 0.01$. Later we managed to tune the LASSO model to an MSE of 0.0563. With randomness we cannot be exact in our estimates, an any indication of numbers herein to four decimal points is just for examination of output from the regressions and by no means an indication of significant digits.

The λ's from Ridge and LASSO are of course just hyper-parameters for tuning to the L2 and L1 norm respectively, and not a parameter directly comparable with each other. For LASSO more time on tuning of the iterative process could improve the result, but the purpose here was only validation. In addition to reviewing the results we spent time testing parts of the code with the Franke function at many different levels of complexity. We also tested for the purpose of finding a suitable size for our real terrain set, which will be the topic for the next section.

For resampling with Ridge we see that higher polynomial degree is regularized with a slightly increased λ. For the bias variance trade offs we see a flatter curve due to the regularization, lowering the variance. This shifts the location of the lowest MSE to the right of the curve with increased λ. For LASSO this is even more pronounced, as the L1 regularizaton of LASSO at some point performs variable selection, setting some coefficients to zero. This flattens the variance and the corresponding bias-variance trade off plot becomes flatter. We have provided bias-variance trade off plots with the λ that gave the lowest MSE.

With a review of these methods we the Ridge method to be the best for the Franke function. We did get marginally better results with LASSO in one run, but we do not consider the difference in the last decimal digits significant. Ridge also includes OLS when $\lambda = 0$ is included, so our preference would be to use Ridge for the Franke function.

3.13 Setup for the Real Terrain Data

3.13.1 Larger Data Set

The size of the terrain data set was 6485401 points which was too large for our laptops. As a consequence, we had to downsize the data set. This problem can be solved in several ways; one could select a small region of the total terrain and only look at that, but this would provide a bad model for the rest of the surrounding terrain. Another way could be to select a set amount of points at random but then one may loose correlation in the terrain. In addition, and the method we opted for, one could select every n-th row and m-th column. We chose $n = m = 50$ giving us roughly $6485401/(50 \cdot 50)$ points. With this procedure we did not get the exact amount for the division, but a total 2701 data points instead. This gives roughly 2161 points in the train set and 540 points in the test set (when using no resampling or bootstrap).

3.13.2 LASSO Parameters

Another difference from the Franke setup was the maximum iterations and tolerance for the LASSO method. Here we sat the maximum iteration to 100 and the tolerance to 0.1. This is due to the larger data set and limited computational power.

3.13.3 Increase in Maximum Polynomial Degrees

We also increased the maximum polynomial degree from 15 to 27 a posteriori after empirical testing. This could be explained by the larger data set that accommodated for a lower variance and thus a higher polynomial degree could be achieved.

4 Results

In the following we describe the results of our regression methods applied on the NASA SRTM DEM terrain data.

4.1 OLS

We see that the lowest MSE for OLS with no resampling was 0.2120 at polynomial degree 16.

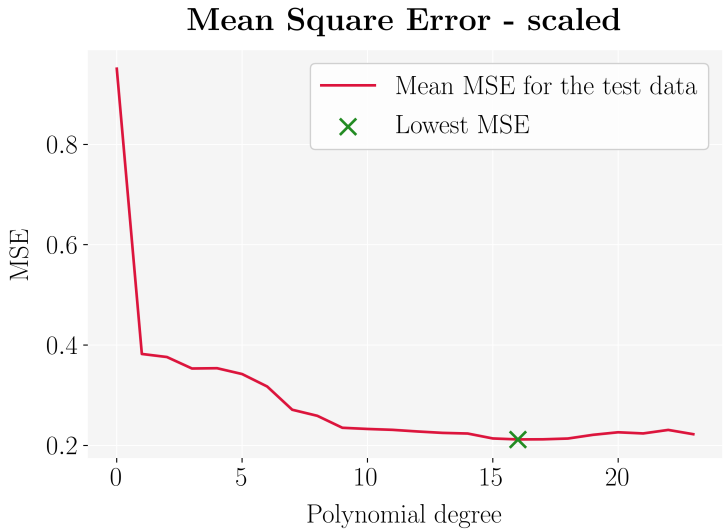


Figure 14: MSE against polynomial degree for OLS. No resampling used.

And the lowest MSE for OLS with cross validation was 0.2152 at polynomial degree 17.

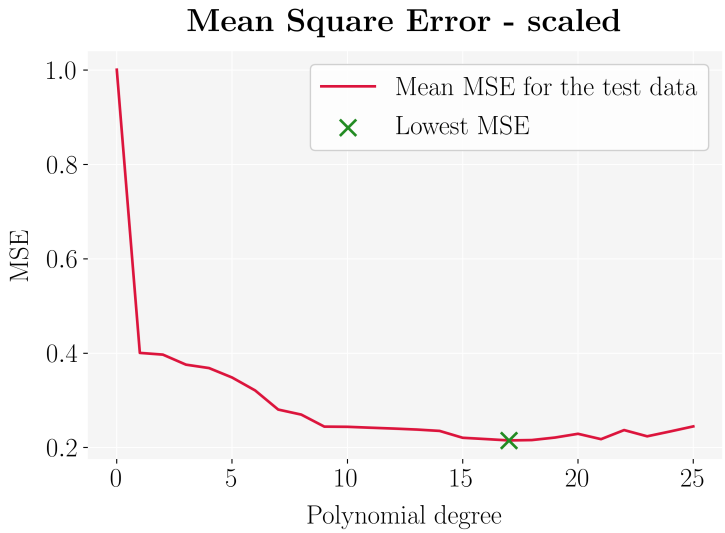


Figure 15: MSE against polynomial degree for OLS. Resampling = cross validation

The highest R^2 score for OLS was = 0.7769 and was found at polynomial degree 16.

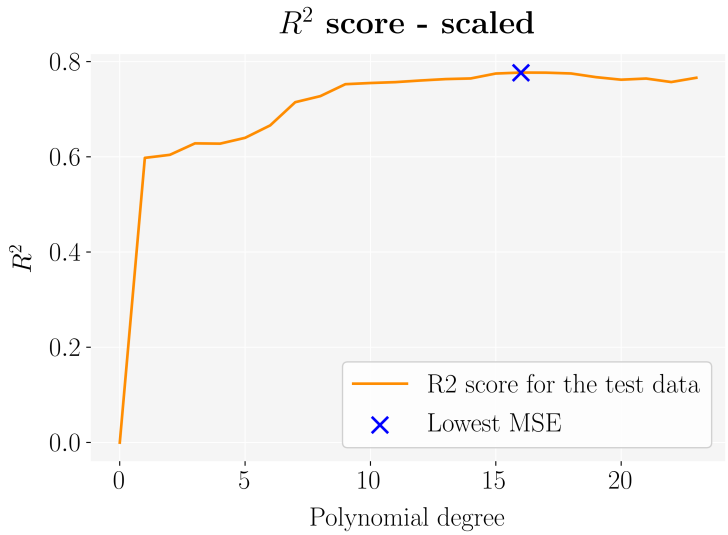


Figure 16: R2 score against polynomial degree for OLS. No resampling used.

The bias variance plot below shows the bias reduction and variance increase for higher complexities. The MSE indicates an u-shaped bias-variance trade off.

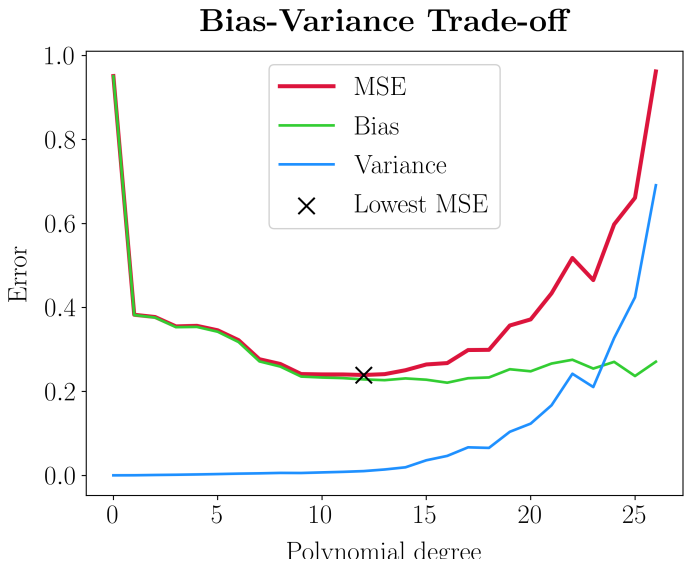


Figure 17: Bias–variance tradeoff for OLS. Resampling = bootstrap.

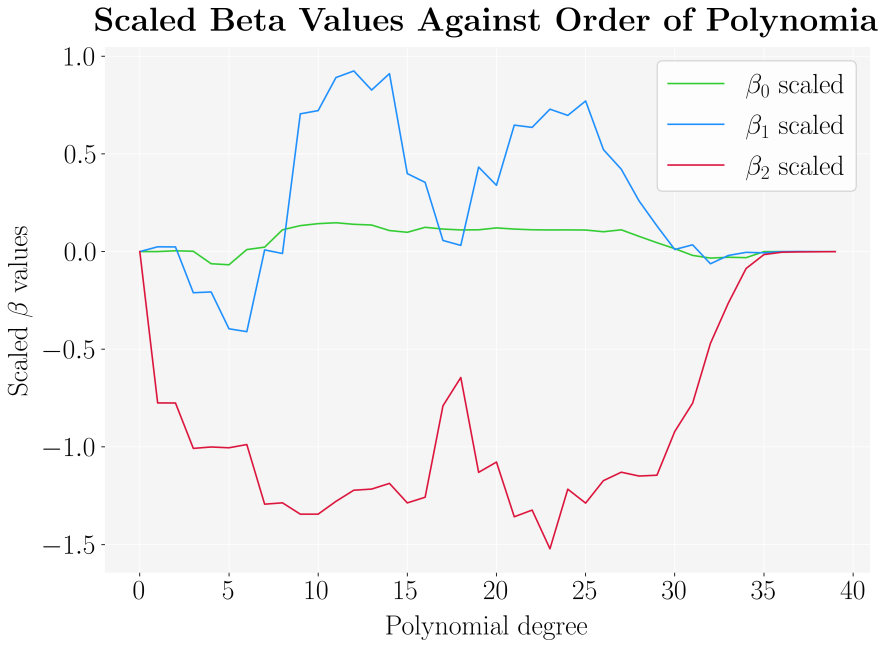


Figure 18: Plot of β_0, β_1 and β_2 as a function of polynomial degree. No resampling used.

4.2 RIDGE

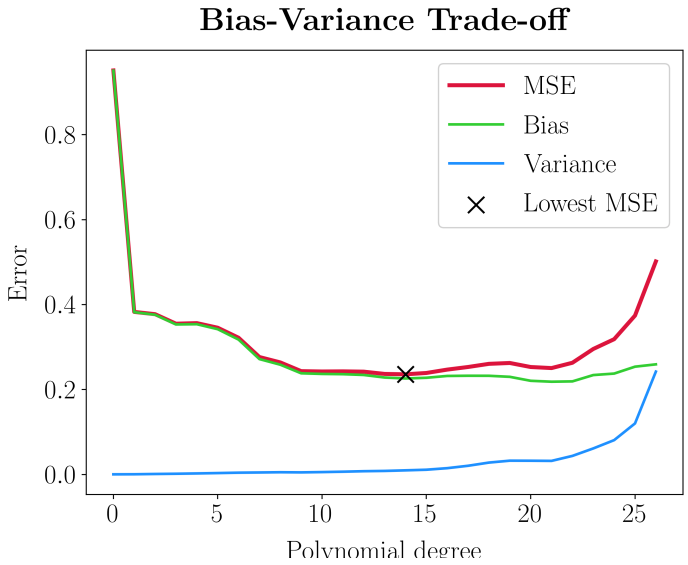


Figure 19: Bias–variance tradeoff for RIDGE. Resampling = bootstrap.

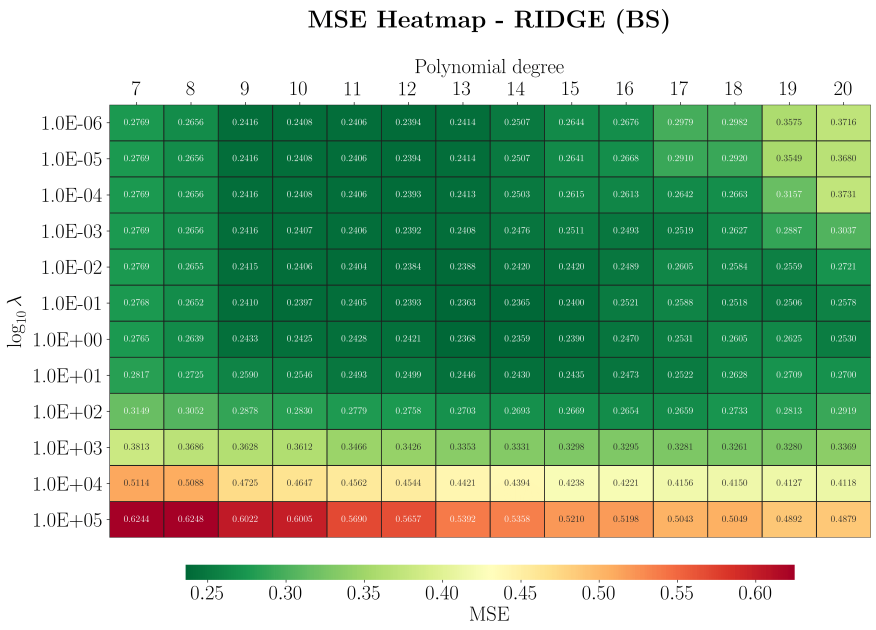


Figure 20: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap

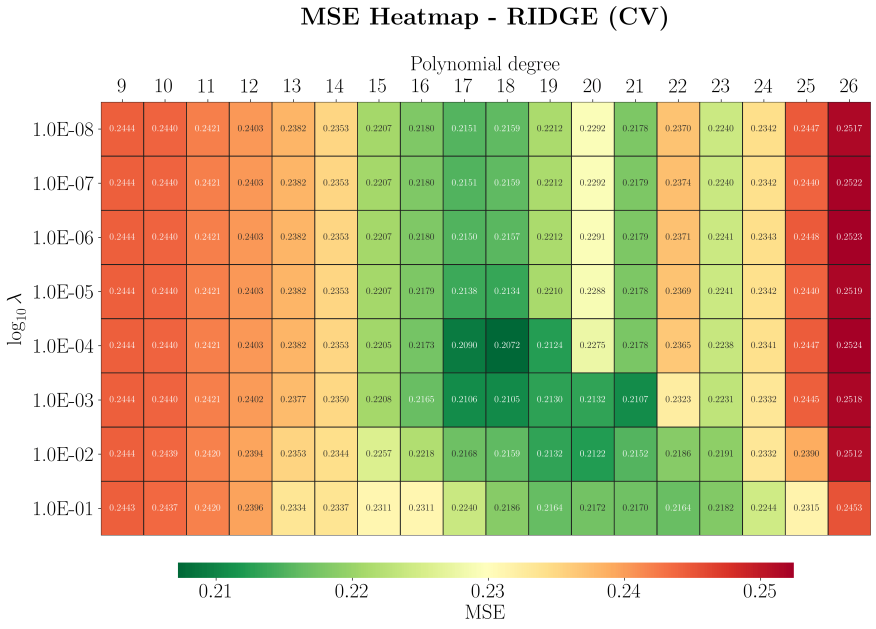


Figure 21: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for RIDGE. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation

4.3 LASSO

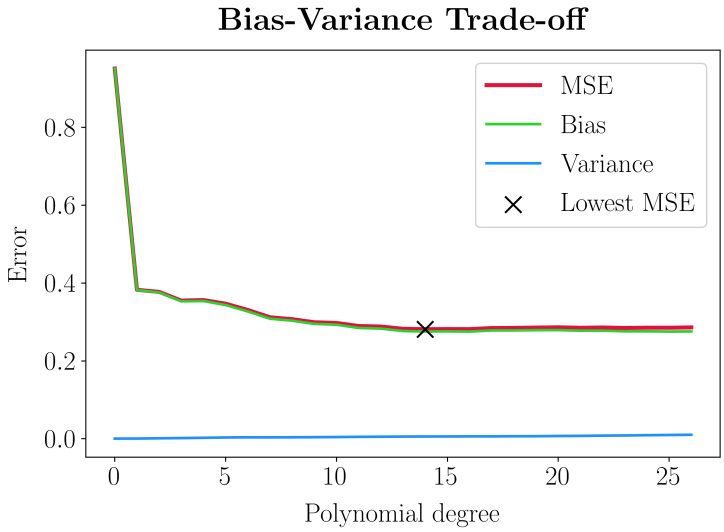


Figure 22: Bias–variance tradeoff for LASSO. Resampling = bootstrap.

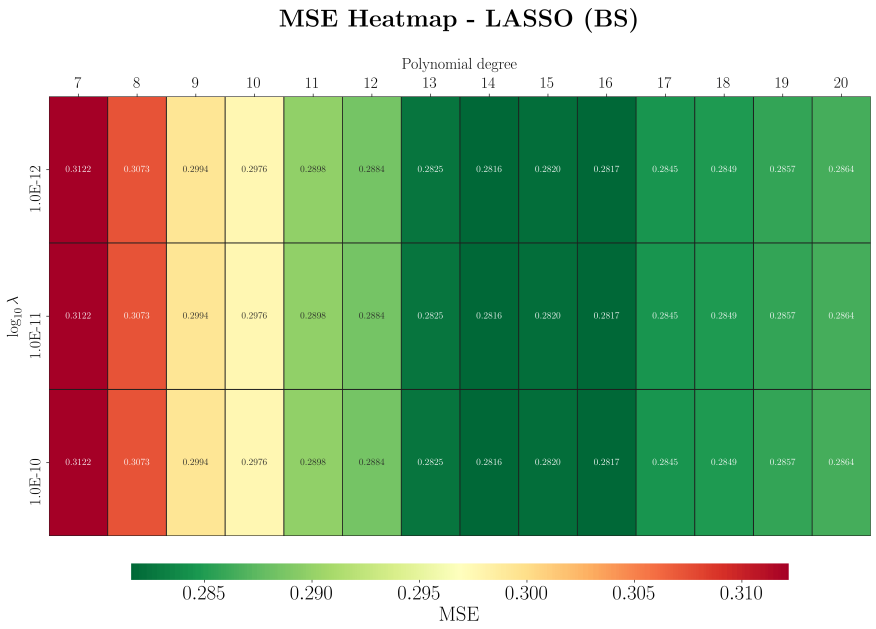


Figure 23: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = bootstrap

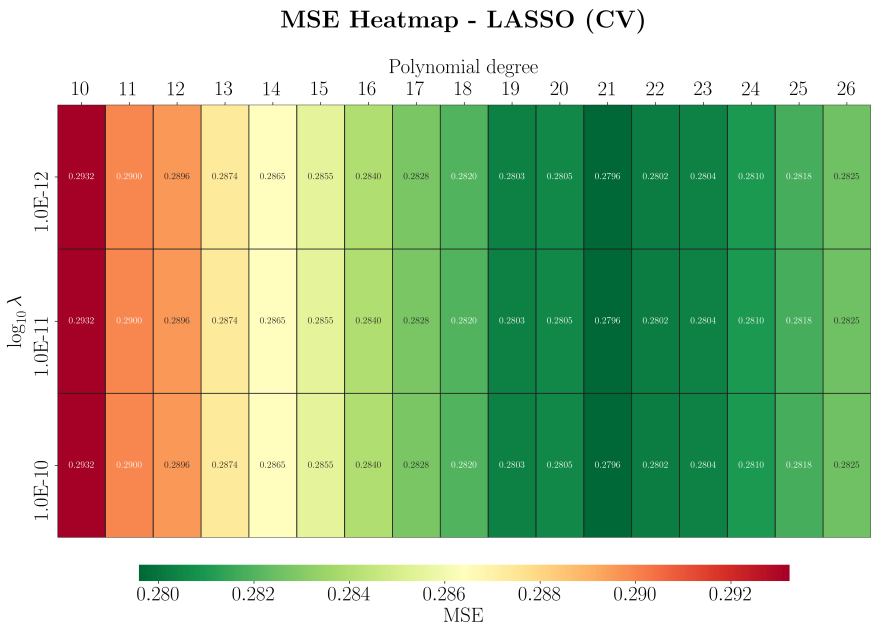


Figure 24: Heatmap plot of MSE as a function of polynomial degree and regularization parameter λ for LASSO. The plot is zoomed in around the polynomial degree and λ that gave the lowest MSE. Resampling = cross validation

4.4 Plots

4.4.1 Optimum Model Against Polynomial Degree and λ

Reg. meth. \ Resamp. model	Bootstrap	Cross Validation	No resampling
OLS	12	17	16
RIDGE	14 (1.0)	18 (1e-4)	-
LASSO	14 (1e-12)	21 (1e-11)	-

Table 1: Optimum model against associated polynomial degree and λ for each regression model and resampling method.

4.4.2 Lowest MSE Achieved

Reg. meth. \ Resamp. model	Bootstrap	Cross Validation	No resampling
OLS	0.2394	0.2152	0.2120
RIDGE	0.2359	0.2072	-
LASSO	0.2816	0.2796	-

Table 2: Lowest MSE against associated polynomial degree and λ for each regression model and resampling method.

5 Discussion

5.1 Bias-Variance Tradeoff

With the down sampled terrain data we had 2701 data points, a larger set compared to the 400 data points we used during validation. We see that this size on the terrain data allows the bias-variance tradeoff to occur at higher polynomial degrees. We tested other partitions of the data set from 400 to 60k data points, and explored how this shifted the min MSE location. In the end we found the current size of 2701 the most interesting, enabling us to explore the effects of regularization at high polynomial levels still attainable with our limited laptop computer resources. At around polynomial degree of 30 we would approach 0.18 predictors per data point, which was enough to trigger over fitting in our plots. We saw from our tests of the Franke Function an optimum polynomial degree of 8 with Ridge CV and this translates to a ratio of .11 coefficients per data point. Considering that the Franke function is a smooth function we expected overfitting to occur earlier on the terrain data, and with Ridge CV we get an optimum polynomial degree of 18. This is equivalent to a ratio of .07 coefficients per data point. We also see that the lowest MSE values we get for the terrain data are higher than the lowest achieved for the Franke function. This is in line with our expectations as discussed in the Theory section.

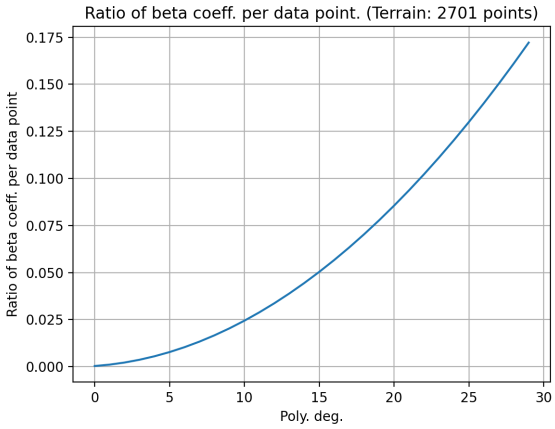


Figure 25: Ratio of beta coeff versus data points per polynomial degree.

In the OLS bias-variance plot we see the expected effect of overfitting as we pass polynomial degree 12. At this point we have reduced the bias but variance starts to build up. At higher degrees the variance dominates and we have clear overfitting. For the Ridge bias-variance plot we see a dampening effect due to regularization. At very high degrees we still get a large increase in variance and MSE. As mentioned in the theory part Ridge does not make any selection, so the higher order polynomials will now at a certain point cause the variance to increase.

Finally we see that the LASSO bias-variance plot shows a flat MSE curve beyond a certain threshold. At this point LASSO has probably performed a selection of variables, such that the higher order variables are excluded.

5.2 Predictor Coefficients (betas)

We have plotted the first three predictor coefficients β_0, β_1 and β_2. Here we observe that the β_0 coefficient, the constant term for the intercept, stays somewhat stable. The intercept just shifts the terrain approximation up or down and as we introduce higher order polynomials, they will take over the explanatory power, in the same way a Taylor expansion of a curve allows it to be described with higher degree coefficients. As we move to higher degrees we see that β_1 and β_2 coefficients, for the independent variable x and y respectively, experience larger fluctuations. The

coefficient for the first ordered independent variable x, β_1, seems to favour a positive sign, while the coefficient for the other first ordered independent variable y, β_2, seems to tend negative. This could indicate the overall slope of the terrain surface along these axis.

5.3 Heatmaps and Model Selection

We consider the results from our 10-fold CV to be the most suitable method for model selection. There is a bias-variance trade off with the choice of k in a CV, and for our large data set we chose k=10. 10-fold. While CV may underestimate the true test MSE, it is very good at finding the correct level of flexibility (James et al. 2013. Bootstrap results can then be compared for test MSE. For OLS the lowest MSE was attained at a polynomial degree of 17 with CV. With regularization using either Ridge or LASSO we tried to improve this result further. The associated heat maps provide insight to the optimum combination of polynomial degree and λ, but also the expected magnitude of selecting a different and perhaps simpler model than the theoretical optimum. For the Ridge heatmap plots, we see in general that higher polynomial order require stronger regulation with a larger λ. This is as expected, ridge allows the inclusion of higher degrees, since their coefficients will be dampened. The heatmap thus have a stepwise pattern, as we move to the right from lower to higher polynomial degrees the ridge optimization of the loss function enforces a stronger regularization of the betas moving down in a stair like pattern. Ridge with CV gives an optimal model with polynomial degree 18 and $\lambda = 1e - 04$. With our Ridge BS we get a lower complexity at polynomial degree 14 and $\lambda = 1.0$. It should be noted that we kept a single test set outside of the entire bootstrap loop (we did not bootstrap the test-train split), and this could affect the bootstrap to arrive at a lower optimum polynomial degree, while the 10-fold cross validation arrives at the best optimum MSE after testing for 10 distinct test sets.

LASSO with cross validation arrives at an optimum polynomial degree of 21, but as noted before, LASSO may force some betas to zero. This explains that there is very little difference in MSE if we select a simpler model. We got the lowest test MSE using Ridge on the terrain data. In terms of model complexity the chosen Ridge model should not have a complexity higher than the optimum at 18. However, as detailed in the Theory section, a different K-fold split could get other results, so even a simpler model should be considered.

5.4 Scaling Issues

As noted in the Method section we got unreliable results when we tried to scale the design matrix. We therefore performed scaling at an early pre-processing stage of our code. This is a topic we will investigate further for other projects.

5.5 Computational restrictions

We ran in to constraints with our laptop computers. A MacBook Pro with 16GB M1 was used, but not sufficient to fully explore the opportunities with LASSO regression. We noted that we could have taken steps to optimize our code for speed, perhaps vectorizing more of the code and by converting some of the more intensive tasks into C++. We compared both SVD and QR algorithms and realised that for larger data sets some prior calculations of the run time is required, as well as to plan longer code executions during night time. The authors have now gained access to HPC (High Performance Computing) facilities which may prove relevant for future project work.

6 Conclusion

In this project we have studied the linear regression methods OLS, Ridge and LASSO with the aim to fit a n-th order polynomial function to the data. We have tested the fit measuring test-MSE on unseen data, and we have applied resampling techniques. After validation of the methods on the Franke function we analysed real terrain data. We found Ridge regression to perform best on the terrain data, but noted that the overall fit was not impressive. We saw that the bias-variance trade off occurred at an earlier stage with the terrain data when measuring the ratio of coefficients per data point. We also got a larger MSE (worse fit) on the terrain data, compared to our fit on the smooth Franke function. Terrain data is highly irregular and it is evident that (x, y)-coordinates on a surface spanning a large area may be a poor predictor of elevation z, especially for certain geographic regions. Instead of a global function acting on the entire space, it could be advisable to use splines, splitting the area of interest in smaller sub-regions and make regression on each sub-region. As a concluding remark, estimations of the true test MSE and associated regression coefficients involves uncertainties. It was beyond the scope for this project to provide confidence intervals for each statistic in this report so the reader is advised to not treat them as exact point estimates but rather as values serving the main purpose of this project, the study of regression methods.

Bibliography

Azzalini, Adelchi and Bruno Scarpa (2012). *Data analysis and data mining: An introduction*. OUP USA.

Box, George EP (1979). 'Robustness in the strategy of scientific model building'. In: *Robustness in statistics*. Elsevier, pp. 201–236.

Efron, Bradley and Robert J Tibshirani (1994). *An introduction to the bootstrap*. CRC press.

Hastie, Trevor, Robert Tibshirani and Martin Wainwright (2015). 'Statistical learning with sparsity'. In: *Monographs on statistics and applied probability* 143, p. 143.

Hoerl, Arthur E and Robert W Kennard (1970). 'Ridge regression: applications to nonorthogonal problems'. In: *Technometrics* 12.1, pp. 69–82.

Höfer, Thomas, Hildegard Przyrembel and Silvia Verleger (2004). 'New evidence for the theory of the stork'. In: *Paediatric and perinatal epidemiology* 18.1, pp. 88–92.

James, Gareth et al. (2013). *An introduction to statistical learning*. Vol. 112. Springer.

Lay, David C, Steven R Lay and Judi J McDonald (2016). *Linear algebra and its applications*. Pearson.

Lindstrøm, Tom L (2017). *Spaces: An introduction to real analysis*. Vol. 29. American Mathematical Soc.

Plackett, Robin L. (1972). 'Studies in the History of Probability and Statistics. XXIX The discovery of the method of least squares'. In: *Biometrika* 59, pp. 239–251.

Santosa, Fadil and William W Symes (1986). 'Linear inversion of band-limited reflection seismograms'. In: *SIAM Journal on Scientific and Statistical Computing* 7.4, pp. 1307–1330.

Süli, Endre and David F Mayers (2003). *An introduction to numerical analysis*. Cambridge university press.

Tibshirani, Robert (1996). 'Regression shrinkage and selection via the lasso'. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288.

Appendix

A Github Repository

The code and associated test runs for this project is available at:

https://github.com/JouvalSomer/FYS-STK3155