UiO : University of Oslo

FYS-STK4155 – Applied Data Analysis and Machine Learning

# Project 1: Regression Analysis and Resampling Methods

October 10, 2020


*Authors:*

Håkon Berggren Olsen

Elisaveta Dombrovski

Lasse T. Keetz

*Lecturer:*

Prof. Morten Hjorth-Jensen

**Abstract**

Making predictions on data and finding correlations between different input variables has high relevance across scientific disciplines, as well as for a plethora of non-scientific applications. Here, the field of machine learning provides state-of-the-art techniques. Linear regression is one popular machine learning approach that aims to fit a continuous function to a discrete data set. It is capable to make predictions for new input data, to unravel general trends, and to evaluate the importance of individual input parameters. In this report, we are presenting three frequently used linear regression techniques: Ordinary Least Squares, ridge regression, and Lasso regression. We touch upon their underlying theories and, firstly, we observe their performances to reproduce a known two-dimensional function (Franke function) and, secondly, we apply them to model real terrain data. We use the Mean Squared Error and the $R^2$-score to evaluate our models. Furthermore, we investigate the implications of using two common resampling methods, namely the bootstrap method and K-fold cross-validation, for model evaluation. In both use cases, linear regressions are able to provide good fits for the data sets. However, we find that Ordinary Least Squares and Ridge regression perform better than Lasso regression in our study settings, even though we conclude that this shall not be generalized for other cases. Moreover, it is generally important to state that even though we are able to reproduce the real terrain data well in our study, it is always important to critically evaluate the predictive power and applicability of machine learning methods.

# Contents

# 1 Introduction

Machine learning is a discipline in computer sciences that gained a lot of popularity among various scientific fields within the last two decades. Even though many of its fundamental principles, for instance the Bayes' Theorem or the Least Square Methods, have been discovered more than two hundred years ago, only the recent increase in computational power and the access to large data sets made the effective application of machine learning possible. The very underlying principle of every machine learning algorithm is the process of gaining knowledge from experience. In contrast to statistics, where we are generally rather interested in finding probability distributions within a given data set, in machine learning, we aim to recognize correlations between parameters of data sets to subsequently predict potential future outcomes with fresh input data. Especially when dealing with large multidimensional data sets, machine learning can help to identify the individual relations between each features and the outcome. A classic example is the study of human genes and their individual linkages to certain diseases (e.g. 3; 4).

Machine learning algorithms can be roughly subdivided into three categories with different purposes, namely regression, classification and clustering (6). In regression, we are searching for a multidimensional function or model that resembles our given output data well enough in a way that we can later use this model to make statements about the output of new input data. An example of regression is the prediction of house prizes based on selected house features, such as the number of rooms, the wealth of the neighborhood or the index of accessibility to radial highways (2). In classification, on the other hand, we are interested in the automatized categorization of input data into previously defined classes that hold certain features. An often referred to classification example is the correct identification of handwritten numbers from zero to nine (10). Finally, with clustering, we can categorise input data, even without having to define the number or the characteristics of classes beforehand. An example of clustering could be to assign students to different learning groups based on their performance in school without knowing the overall performance of the students in the class. Because in clustering we cannot monitor its performance while training the algorithm, it is therefore considered to be an unsupervised machine learning method, while regression and classification are supervised algorithms.

In principle, all machine learning algorithms follow a similar structure of using a subset of 2/3 to 4/5 of some available data for developing or training a model that links the input data with the output data. The model is then verified against the remaining subset of test data that has not been included during the training period - therefore, it is often assumed to be independent. By analyzing the error between our established model and the true output values of the test data subset, we can evaluate the performance of our model. In those cases, where enough data is available, it can be advisable to consider using a third validation subset, which can help to improve the model before its final evaluation with the test data (7).

In this report, we will focus on the application of linear regression. In linear regression we aim to fit a continuous likelihood function $P(\mathbf{y}|\mathbf{x})$ that links a set of input data $\mathbf{x}$ (also referred to as independent, predictor, or explanatory variables) to a given set of output data $\mathbf{y}$ (also referred to as dependent, outcome, or response variables) in a linear proportion. Hence, we can determine causal dependencies and relationships to predict future outcomes of fresh data and approximate interpolations between existing data points. Linear regression is often used as a gateway into the realm of machine learning. On one hand, it is relatively simple to implement, and on the other hand, it introduces fundamental and ubiquitous machine learning principles, such as cost functions, the bias-variance tradeoff and resampling methods. Unlike other techniques, it also provides analytical solutions for the parameters $\beta_j$, mean values, variances and confidence intervals.

In general, different methods to perform linear regression analysis exist. In this report, we explain the underlying theory and differences behind three methods with an emphasis on their individual influence on the algorithm performance. The performances are subsequently evaluated, first, by fitting a model to an a priori known two-dimensional function, the Franke Function, and, thereafter, on a second set of real terrain elevation data, where the real underlying distribution is unknown.

The detailed structure of the report is as follows. In Chapter 2 we will provide the reader with the basic assumptions and mathematics behind three different linear regression techniques. Thereafter, in Chapter 2.1.1, we present two cost functions that can be used to evaluate the performance of an established model against "true outcome" data. In Chapters 2.1.3, 2.1.4 and 2.1.5 we present the following linear regression techniques: Ordinary Least Squares, Ridge regression, and Lasso regression. We discuss their characteristics, differences, advantages, and limitations. With regards to their analytical solutions, the following chapter 2.1.6 presents singular value decomposition, a method to render singular matrices invertible.

Subsequently, we will define and explain the Bias-Variance trade off in modelling (ch. 2.2) and show how to determine confidence intervals within linear regression (ch. 2.3). Chapter 2.4 presents two common data resampling techniques to improve model training and evaluation, namely bootstrapping and K-fold cross-validation. Subsequently, in chapter 3, we will present the methodology of our model experiments that apply the presented theories - specifically, they are tested against the different linear regression implementations, different resampling techniques, and evaluation procedures. Relevant results will be presented and discussed in chapter 4, where we also briefly reflect on the pitfalls of machine learning applications. The final chapter 5 summarizes the insights we deem most important.

# 2 Theory and Methods

As mentioned, linear regression is often introduced at early stages of machine learning courses due to its relatively simple and intuitive implementation. The following subsections provide the reader with some basic assumptions and mathematics behind different linear regression techniques as well as a selection of common evaluation and training approaches in machine learning.

## 2.1 Linear Regression

Assume that we have a data set $\boldsymbol{D}$ that includes a set of input data $\boldsymbol{x}$ and a set of output data $\boldsymbol{y}$. The relationship between the input and the output data can be described as

$$\boldsymbol{y} = f(\boldsymbol{x}) + N(\mu, \sigma^2), \tag{1}$$

where $f(\boldsymbol{x})$ represents an unknown underlying function depending on $\boldsymbol{x}$ and $N(\mu, \sigma^2)$ is normally distributed noise around a value $\mu$ with the variance $\sigma^2$. In linear regression we assume that the noise mean $\mu$ equals 0. Furthermore, we state that $f(\boldsymbol{x})$ can be approximated by a model $\tilde{\boldsymbol{y}}$. Since we usually deal with a discrete type of data set, equation 1 can now be rewritten as

$$\boldsymbol{y}(\boldsymbol{x_i}) \simeq \tilde{\boldsymbol{y}}_{\boldsymbol{i}}(\boldsymbol{x_i}) + \epsilon_i, \tag{2}$$

where $\boldsymbol{y}(\boldsymbol{x_i})$ are our discrete output values, $\tilde{\boldsymbol{y}}_{\boldsymbol{i}}(\boldsymbol{x_i})$ is a model depending on our discrete input values $\boldsymbol{x_i}$ and $\epsilon_i$ is normally distributed noise around 0, that is $\epsilon_i = N(0, \sigma^2)$.

Linear regression fits a linear model of the $nth$ degree of $\boldsymbol{x}$ to $\boldsymbol{y}$. There is a linear statistical relationship

between the regression function and the set of unknown variables $\boldsymbol{\beta}$ that are approximated from our data set. If we look at the case where we only have a one dimensional input parameter set $\mathbf{x}$, our model could be represented by a first order equation of the form

$$\tilde{\boldsymbol{y}}_{\boldsymbol{i}} = \beta_0 + \beta_1 \boldsymbol{x}_{\boldsymbol{i}}, \tag{3}$$

where $\beta_0$ represents the intersect of the linear fit with the $\boldsymbol{y}(\boldsymbol{x}_{\boldsymbol{i}})$ axis and $\beta_1$ represents the slope of the line. Our goal in Linear Regression is it to find the optimal $\hat{\boldsymbol{\beta}}$ that links the input and the output data in a way that minimizes the deviation of our model from the true training output data as well as for the test data. Since we usually do not know the complexity of the underlying function, we can often start with a simple first order approximation for our model and add more complexity to it later.

As the input data sets in machine learning are often multidimensional, in other words, there is more than one input feature, we replace $\boldsymbol{x}$ in equation 1 and equation 2 with a respective matrix notation $\boldsymbol{X}$. The input data set $\boldsymbol{X}$, also known as the feature matrix or the design matrix, has the following generalized form:

$$X = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \dots & x_{0,p-1} \\ x_{1,0} & \ddots & & & \vdots \\ x_{2,0} & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ x_{n-1,0} & \dots & \dots & \dots & x_{n-1,p-1} \end{bmatrix}, X \in \mathbb{R}^{n \times p}. \tag{4}$$

In the special case of quadratic symmetric problems, $\boldsymbol{X}$ becomes

$$X = \begin{bmatrix} 1 & x_0^1 & x_0^2 & x_0^3 & \dots & x_0^{n-1} \\ 1 & x_1^1 & \ddots & & & \vdots \\ 1 & x_2^1 & & \ddots & & \vdots \\ 1 & \vdots & & & \ddots & \vdots \\ 1 & x_{n-1}^1 & \dots & \dots & \dots & x_{n-1}^{n-1} \end{bmatrix}, X \in \mathbb{R}^{n \times n}. \tag{5}$$

and, thus, equation 2 becomes

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} = \begin{bmatrix} 1 & x_0^1 & x_0^2 & x_0^3 & \dots & x_0^{n-1} \\ 1 & x_1^1 & \ddots & & & \vdots \\ 1 & x_2^1 & & \ddots & & \vdots \\ 1 & \vdots & & & \ddots & \vdots \\ 1 & x_{n-1}^1 & \dots & \dots & \dots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{n-1}, \end{bmatrix} \tag{6}$$

where every single outcome data point is then described as

$$\begin{aligned}
y_0 &= \beta_0 x_{0,0} + \beta_1 x_{0,1} + \beta_2 x_{0,2} + ... + \beta_{n-1} x_{0,n-1} + \epsilon_0, \\
y_1 &= \beta_0 x_{1,0} + \beta_1 x_{1,1} + \beta_2 x_{1,2} + ... + \beta_{n-1} x_{1,n-1} + \epsilon_1, \\
y_2 &= \beta_0 x_{2,0} + \beta_1 x_{2,1} + \beta_2 x_{2,2} + ... + \beta_{n-1} x_{2,n-1} + \epsilon_2, \\
&.... \\
y_i &= \beta_0 x_{i,0} + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_{n-1} x_{i,n-1} + \epsilon_i, \\
&.... \\
y_{n-1} &= \beta_0 x_{n-1,0} + \beta_1 x_{n-1,1} + \beta_2 x_{n-1,2} + ... + \beta_{n-1} x_{n-1,n-1} + \epsilon_{n-1}.
\end{aligned} \tag{7}$$

### 2.1.1 Cost Function

The cost function, also referred to as error function or loss function, is used to measure how much a model $\tilde{\boldsymbol{y}}$ differs from the given output data set $\boldsymbol{y}$. Since we are seeking for an optimal $\hat{\boldsymbol{\beta}}$ that makes the difference between the model and the outcome data rather small, finding $\hat{\boldsymbol{\beta}}$ becomes a minimization problem of our cost function. One commonly used cost function in statistics is the so called relative error, which is described as

$$C(\boldsymbol{\beta}|\boldsymbol{x})_{RE} = \left| \frac{\boldsymbol{y} - \tilde{\boldsymbol{y}}}{\boldsymbol{y}} \right|. \tag{8}$$

However, since the relative error is an absolute value, its derivative cannot be solved analytically and is therefore rarely used in machine learning. Instead much more often the Mean Squared Error is used, which is defined as

$$C(\boldsymbol{\beta}|\boldsymbol{x})_{MSE} = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2. \tag{9}$$

We will use $C(\boldsymbol{\beta}|\boldsymbol{x})_{MSE}$ as the cost function in this report and, hence, define $C(\boldsymbol{\beta}|\boldsymbol{x}) = C(\boldsymbol{\beta}|\boldsymbol{x})_{MSE}$ for simplicity. The optimal $\hat{\boldsymbol{\beta}}$ then becomes

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} C(\boldsymbol{\beta}|\boldsymbol{x}). \tag{10}$$

### 2.1.2 R2 Score

The $R^2$ score is defined by

$$R^2 = 1 - \frac{\sum_{i=0}^{n-1} (\tilde{y}_i - y_i)^2}{\sum_{i=0}^{n-1} (\tilde{y}_i - \bar{\tilde{y}})^2} \tag{11}$$

Where $\bar{\tilde{y}}$ is the mean of $\tilde{\boldsymbol{y}}$. The $R^2$ is defined between $R^2 \in [0, 1]$ and indicates correlation between the predicted fit and the data. Here, a value close to 1 indicates high correlation, while a value close to 0 represents low correlation.

### 2.1.3 Ordinary-Least-Squares

In Ordinary-Least-squares (OLS), we are interested in finding the optimal parameter $\hat{\boldsymbol{\beta}}_{\boldsymbol{OLS}}$ that minimizes the difference between our model $\tilde{\boldsymbol{y}}_{\boldsymbol{i}}$ and our training data set $\boldsymbol{y}_{\boldsymbol{i}}$. The cost function for OLS is defined as

$$C(\boldsymbol{\beta}|\boldsymbol{x})_{OLS} = C(\boldsymbol{\beta})_{OLS} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\}. \tag{12}$$

After setting the first derivative of $C(\boldsymbol{\beta}|\boldsymbol{x})_{OLS}$ to zero, we end up with an analytical solution for $\hat{\boldsymbol{\beta}}_{\boldsymbol{OLS}}$ that is

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{13}$$

Even though the OLS approach provides an easy solution at first glance, it is important to note that it comes with the condition that the term $(\boldsymbol{X}^T\boldsymbol{X})$ is invertible. By definition, a $n \times n$ matrix $\mathbf{A}$ is invertible if and only if there is a $n \times n$ matrix $\mathbf{B}$ that multiplied with $\mathbf{A}$ results in the identity matrix $\mathbf{I}$. A square matrix h that is not invertible is called a singular matrix and is characterized by its determinant equaling 0. The problem of inverting a singular matrix can be bypassed by employing a regularization parameter $\lambda$, which is done in ridge regression and Lasso regression (chapter 2.1.4 and 2.1.5, respectively), or by computing the pseudoinverse with singular value decomposition (chapter 2.1.6). Another often occurring problem when employing the OLS method is overfitting. The term overfitting refers to a model being well adapted to its training data set while lacking the necessary generality to also perform well on the test data set. We will again touch upon the topic of overfitting in chapter 2.2

### 2.1.4 Ridge Regression

In the previous section, we have shown how an optimal parameter set $\hat{\boldsymbol{\beta}}$ can be obtained with the OLS method. However, this comes with the condition that the term $(X^TX)^{-1}$ must be solvable, which is only the case for non-singular matrices. To avoid this problem, ridge regression, also called L2-norm regularization, utilizes a regularization parameter $\lambda$ that is added to the diagonal of the original feature matrix $(X^TX)$ and,

thereby, makes any square matrix invertible. $\hat{\boldsymbol{\beta}}_{\boldsymbol{Ridge}}$ then becomes

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{Ridge}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}, with\ \lambda \geq 0. \tag{14}$$

Note that if $\lambda = 0$ then $\hat{\boldsymbol{\beta}}_{\boldsymbol{Ridge}} = \hat{\boldsymbol{\beta}}_{\boldsymbol{OLS}}$. In addition to circumventing the invertibility issue described above, it was later discovered that adding a parameter $\lambda$ also helps to avoid overfitting. For instance, if we take a point cloud of rather scattered $\mathbf{y}$ values and fit a model through it, the single parameters $\beta_j$ can show a large scatter of positive and negative values that are described as

$$P(\beta_j)_{Ridge} = \prod_{i=0}^{p-1} N(0, \tau^2) = e^{-\beta_j^2/2\tau^2}, \tag{15}$$

where $P(\beta_j)$ resembles a Gaussian distribution with a mean value $\mu = 0$ and a variance $\tau^2$. Additionally the maximum likelihood estimator (MLE) states that each $y_i$ can be seen as an independent stochastic variable that is given by the normal distribution

$$MLE : P(y_i|x_i\boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i-\mu)^2/2\sigma^2} \propto e^{-(x_i-\mu)^2/2\sigma^2}. \tag{16}$$

In the following, we are making use of Bayes' Theorem, which links the probability of the occurrence of event A given that event B already happened to the probability of the occurrence of event B given that event A already happened. Accordingly, it states

$$P(A|B) \propto P(B|A)P(A). \tag{17}$$

By combining the MLE from equation 16 and the principles from the Bayes' Theorem from equation 17 we can infer

$$\begin{aligned} P(\boldsymbol{\beta}|\boldsymbol{X}\boldsymbol{y}) \propto P(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\beta})P(\boldsymbol{\beta}) &= \prod_{i=0}^{n-1} P(y_i|X_i\boldsymbol{\beta}) \prod_{j=0}^{p-1} P(\beta_j)_{Ridge} \\ &= \sum_{i=0}^{n-1} log\ P(y_i|X_i\boldsymbol{\beta}) + \sum_{j=0}^{p-1} log\ P(\beta_j)_{Ridge} + C \\ &\propto \sum_{i=0}^{n-1} (y_i|X_i\boldsymbol{\beta})^2 + \lambda \sum_{j=0}^{p-1} \beta_j^2, \end{aligned} \tag{18}$$

With C representing a constant. From this we conclude that

$$\lambda \propto \frac{1}{2\tau^2}, \tag{19}$$

which means that an increase of the regularisation variable $\lambda$ results in the reduction of the variance in $P(\beta_j)$ (see equation 15).

Intuitively, an increase of $\lambda$ can be understood as a reduction of the steepness of the slope of a model. If we only have a restricted amount of data points in our training data set and try to fit a model, chances are that the model that we produce by using OLS (or $\lambda = 0$) and high complexity will result in a good fit to the training data, i.e. the MSE will be low. However, even though this model represents our training data points well (i.e. the bias is low, see chapter 2.2), there is a high chance that it will poorly represent the unknown test data set (i.e. the variance is high). Introducing $\lambda$ to the term causes the variance of $\boldsymbol{\beta}$ to shrink. Consequently, the model curve is "flattened" - or in other words, the model output gets less sensitive to the input values of X. The new model will have a higher MSE regarding the training data (increased bias) but it might also resemble new test data better (reduced variance). This is crucial for common machine learning applications, where we are interested in predicting the outcome for new input data. One has $\lambda \geq 0$ where $\lambda = 0$ simply equals OLS, while a large value of $\lambda$ smooths the slopes of our model approaching but never reaching a slope equaling zero. Accordingly, in ridge regression, the regularisation variable or penalty term $\lambda$ shrinks some of the components that are less important (that have small eigenvalues) and thus helps us to avoid overfitting even with models of high complexity. Therefore, we can obtain more generalized models. However, Ridge regression will not allow us to discard input variables completely.

### 2.1.5 Lasso Regression

In consonance with ridge regression, Lasso regression, also referred to as L1-norm regularisation, includes a penalty term $\lambda$. However, contrary to ridge regression, where the probability density function $P(\boldsymbol{\beta})_{Ridge}$ was given by a Gaussian distribution (see equation 15), Lasso regression incorporates a Laplace distribution that is

$$P(\boldsymbol{\beta}|\boldsymbol{X}\boldsymbol{y})_{Lasso} \propto \prod_{i=0}^{n-1} P(y_i|X_{i*}\boldsymbol{\beta}) \prod_{j=0}^{p-1} e^{-|\beta_j|/\tau}. \tag{20}$$

After minimizing the cost function we end up with

$$\hat{\boldsymbol{\beta}}_{Lasso} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \frac{1}{n}\sum_{i=0}^{n-1}(y_i - X_{i*}\boldsymbol{\beta})^2 + \lambda\sum_{j=0}^{p-1}|\beta_j|. \tag{21}$$

Note that in contrast to equation 18 for ridge regression, where we were dealing with $\lambda \sum_{j=0}^{p-1} \beta_j^2$, in Lasso regression, we have $\lambda \sum_{j=0}^{p-1} |\beta_j|$. This complicates finding an analytical solution. Moreover, contrary to ridge Regression, where components can be shrunk but never brought to zero, Lasso regression is able to completely disregard less important components and, therefore, reduce the dimensions of the domain. For a one linear model, this is visualized in Figure 1, where $\hat{\boldsymbol{\beta}}$ marks the optimal parameters for $\beta_1$ and $\beta_2$ how it would be chosen with the OSL method. The red circles around $\hat{\boldsymbol{\beta}}$ mark steady MSE values. With regularization we define a specific value t that restricts the values that our parameters $\beta_1$ and $\beta_2$ can take, with $\lambda \sum_{j=0}^{p-1} \beta_j^2 t$ and $\lambda \sum_{j=0}^{p-1} |\beta_j| \leq t$, respectively. Since in ridge regression we incorporate a quadratic term, the resulting area (in a 2 dimensional case) in which we allow our $\beta$ parameters to be resembles a circle according to the circle equation $a^2 + b^2 = r^2$ (Figure 1), while Lasso uses a the absolute values of $\beta$, resulting in a diamond shape. In both cases the optimal $\hat{\boldsymbol{\beta}}_{Ridge/Lasso}$ will be situated where the margin of the geometric shape meets the smallest possible MSE. The relationship between $\lambda$ and $t$ depends on the chosen data set.

In case of ridge regression: If the $\hat{\boldsymbol{\beta}}_{OLS}$ is not located exactly on one of the $\beta_{1or2}$ axis (meaning that one of the parameters does not have any influence anyway) then ridge regression is not able to obtain $\beta$ values that are located on an axis either, thus it is never able to completely remove one of the parameters, only to reduce them. In case of Lasso regression: Because of its diamond shape, it can actually obtain a $\hat{\boldsymbol{\beta}}_{lasso}$ that lies on one of the axis, even if $\hat{\boldsymbol{\beta}}_{OLS}$ is not located exactly on one of the $\beta_{1or2}$.

### 2.1.6 Singular Value Decomposition

As stated in Chapter 2.1.3, one problem we face when performing OLS is that we might run into singular matrices that cannot be inverted. The two regularisation techniques we show in the previous chapter, namely ridge and Lasso regression, add a small penalty term $\lambda$ to a square matrix and thus overcome the invertibility problem, while simultaneously preventing overfitting. Another way to invert singular matrices, however, is to compute the pseudoinverse of the matrix by using Singular Value Decomposition (SVD). With SVD, any $m \times n$ matrix $X$ can be decomposed into three matrices so that

$$\boldsymbol{X} = U \Sigma V^T, \tag{22}$$

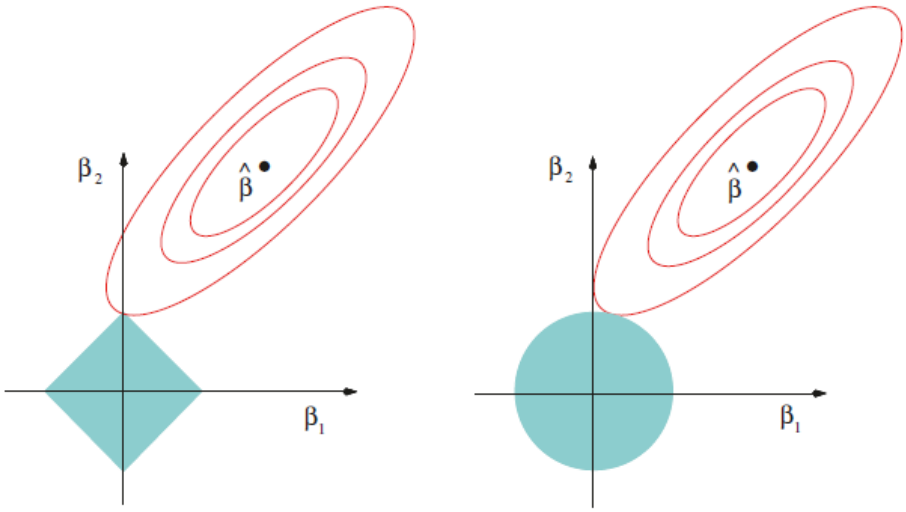



**Figure 1:** *Figure from (5) showing the $L_1$ (left) and $L_2$ (right) constraints, respectively, in blue. The red contour lines represent identical errors with regards to different $\beta$.*

where $U$ has the dimensionality of $m \times m$, $\Sigma$ has the dimensionality of $m \times n$ and $V^T$ has the dimensionality of $n \times n$. In case of a invertible matrix, the pseudoinverse of $X$ is computed as

$$\boldsymbol{X}^{-1} = V\Sigma^{-1}U^T, \tag{23}$$

while in case of a non-invertible matrix, the pseudoinverse of $X$ is computed as (SVD)

$$\boldsymbol{X}^{-1} = (U\Sigma V^T)^{-1} \approx V D_0^1 U, \ with \ D_0^1 = \begin{cases} 1/\sigma_i, & \text{for } \sigma_i > a \\ 0, & \text{otherwise} \end{cases}, \tag{24}$$

with a being a small treshold and $\sigma_i$ being singular values.

## 2.2 Bias-Variance Trade off

The Bias Variance formulas can be shown by rearranging the cost function

$$C(\mathbf{X}, \beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbf{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] \tag{25}$$

Adding and subtracting $E[\tilde{\mathbf{y}}]$ inside the expectancy above

$$\mathbf{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] = \mathbf{E}[(f + \epsilon + \mathbf{E}[\tilde{\mathbf{y}}] - \mathbf{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2] \tag{26}$$

and using the fact that $y = f + \epsilon$, where $\epsilon$ is normally distributed noise centered around mean $\mu = 0$ and with variance $\sigma^2$.

$$= \mathbf{E}[(f - \mathbf{E}[\tilde{\mathbf{y}}])^2 + 2(f - \mathbf{E}[\tilde{\mathbf{y}}])(\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{y}) + \epsilon^2 + (\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2] \tag{27}$$

Where terms with $E[\epsilon]$ are canceled out as the expectancy is zero. Utilizing the linearity of the expected value operator, the equation becomes

$$= \mathbf{E}[(f - \mathbf{E}[\tilde{\mathbf{y}}])^2] + \mathbf{E}[2(f - \mathbf{E}[\tilde{\mathbf{y}}])(\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{y})] + \mathbf{E}[\epsilon^2] + \mathbf{E}[(\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2] \tag{28}$$

The $(f - \mathbf{E}[\tilde{\mathbf{y}}])^2$ terms here are a constant because $f$ is deterministic ($\mathbf{E}[f] = f$), and the expectance is just the argument.

$$= (f - \mathbf{E}[\tilde{\mathbf{y}}])^2 + 2(f - \mathbf{E}[\tilde{\mathbf{y}}])\mathbf{E}[(\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{y})] + \mathbf{E}[\epsilon^2] + \mathbf{E}[(\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2] \tag{29}$$

The expectance $\mathbf{E}[\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{y}]$ becomes $\mathbf{E}[\tilde{\mathbf{y}}] - \mathbf{E}[\tilde{\mathbf{y}}] = 0$. The remaining equation becomes

$$\mathbf{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] = (f - \mathbf{E}[\tilde{\mathbf{y}}])^2 + (\mathbf{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2 + \sigma^2 \tag{30}$$

$$\mathbf{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] = \text{Bias}^2 + \text{Var} + \text{Irreducible Error} \tag{31}$$

The decomposition of the MSE can be shown in Figure 2, where the Bias is high and the variance is low in low complexity (in our case low polynomial degree $n$), and the bias is low and the variance is high in high complexity. We say our model is underfitted if the bias is high and our variance is low. An example can be thought of having polynomial degree $n = 1$ of our model, which would have a low variance since it is a trend line and captures the overall trend of our data, but would not explain the nuances of the data. Alternatively, we say our model is overfitted if the model complexity is high. This happens when we fit the model with a high degree of complexity so that the error of the model on the training data is low, but it generalizes very poorly. The best fit for the data would be to minimize both the bias and the variance, so that it generalizes well with respect to new data but also captures the nuances of the relationships governing the features.

## 2.3 Confidence intervals

We use confidence intervals to estimate the variability of $\boldsymbol{\beta}$. A 95 % confidence interval for $\beta_{OLS}$ is defined as

$$CI_i = \beta_i \pm 1.96\sqrt{Var(\beta_i)} \tag{32}$$

The variance of $\beta_i$ can be found by

$$Var(\beta_i) = \sigma^2[(\boldsymbol{X}^T\boldsymbol{X})^{-1}]_{jj} \tag{33}$$

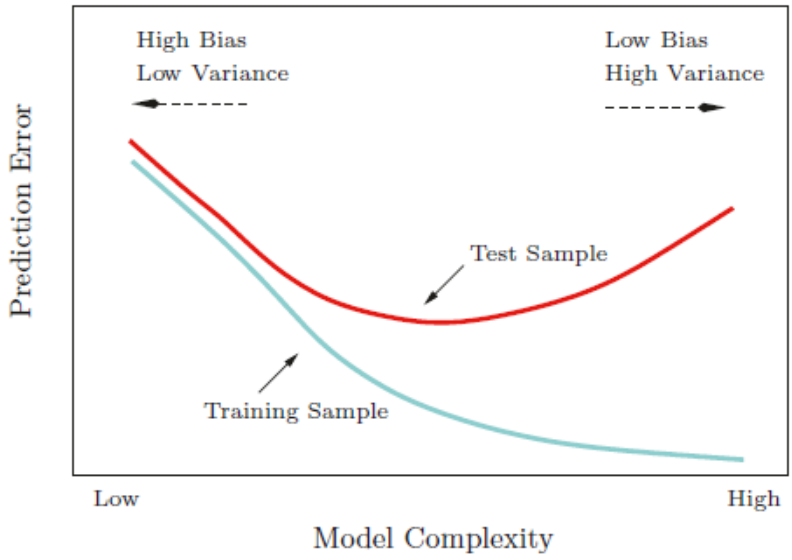



**Figure 2:** *Test and training error as a function of model complexity. The test sample is here the part that is decomposed into bias*$^2$ *and variance. Figure adapted from (5).*

To find the confidence interval for the ridge regression, the variance is for each of the $\beta_i$ is found by taking the diagonal elements of the matrix

$$Var(\beta_i) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X} + \lambda I_{pp})^{-1}\boldsymbol{X}^T\boldsymbol{X}([\boldsymbol{X}^T\boldsymbol{X} + \lambda I_{pp}]^{-1})^T \tag{34}$$

The variances of $\boldsymbol{\beta_{OLS}}$ are always greater than the corresponding variances for beta using ridge regression ($\boldsymbol{\beta_{ridge}}$) (8), as the penalty term induces bias. Computing the variances for the $\beta_{Lasso}$ is possible, but as Lasso regression does not have an analytical solution, it finds the $\beta$ coefficients iteratively and is not in the scope of this report.

## 2.4 Resampling Methods

In machine learning, a given data set is commonly split into a training data set (around 70 %) and a test data set (around 30 %), and sometimes a validation data set. Resampling is a popular technique that allows us to build more robust models and to evaluate their predictive power more realistically. In the following subsection, we will explain the principles and algorithms behind two popular resampling techniques, namely the bootstrap method and the K-fold cross-validation.

### 2.4.1 Bootstrap

Bootstrapping is a resampling technique that can help to approach the true expected value or mean value of the underlying distribution of $\boldsymbol{y}$. It is conducted in several steps:

1. **Draw a new Sample:** From our entire data set $\boldsymbol{D}$, we randomly select a subset of n data points $\boldsymbol{Z_i^*} = (z^*_{j,1}, z^*_{j,2}, z^*_{j,3}, ..., z^*_{j,n})$ B times, with $\boldsymbol{i} = (1, 2, 3, ..., B)$. What is characteristic in the bootstrap method is that the selection of each data point happens with replacement. This means that one data point can be selected multiple times within the one subset $\boldsymbol{Z_i^*}$. In contrast to that, we want to briefly mention another resampling method, which is called Jack-Knive Method. In contrast to bootstrap, the Jack-Knive method employs sampling without replacement and therefore allows a data point $z_{j,i}$ only once in a data subset $\boldsymbol{Z_i^*}$.

2. **Compute the Expected Value:** For every subset $\boldsymbol{Z_i^*}$ we can now compute the expected value $S(\boldsymbol{Z_1^*}), S(\boldsymbol{Z_2^*}), ..., S(\boldsymbol{Z_B^*})$

3. **Compute the Variance of the Expected Values:** The variance of the previously computed expected values give us information about how much our model would change if we used a different data subset and is computed by $var(S(\boldsymbol{Z^*})) = \frac{1}{B-1}\sum_{i=1}^{B}(S(\boldsymbol{Z_i^*}) - \bar{S}^*)^2$, with $\bar{S}^*$ being the sum over all expected value samples.

Bootstrap is a popular method because it guarantees to approach the true variance or expected value of a data set and it is also relatively easy to implement.

### 2.4.2 K-fold cross-validation

K-fold cross validation is another resampling method, that consists of the following steps:

1. Shuffle the entire available data set $\boldsymbol{D}$ randomly.

2. Part the data set into $K$ subsets $D = (d_0, d_1, ..., d_{K-1})$ or folds. Usually, K is a number between 5 and 10.

3. Loop over i = 0:1:K-1

(a) Select subset $d_i$ as your test data $d_{test} = d_i$.

(b) Include every $d_{j\neq i}$ subset in your training data, $d_{training} = \sum d_{j\neq i}$.

(c) Create a model based on your training data.

(d) Compute the error between your model and the test data output $err_i$

4. Compute the total error, $err = \frac{1}{K}\sum_{i=0}^{K-1} err_i$

Choosing large values for K, i.e. creating smaller folds, typically results in lower variance. However, this may lead to an increase in bias. On the other hand, choosing small values for K can lead to high variance (5).

# 3 Methods and Data

## 3.1 Franke Function

In this project, we use the Franke Function to generate the target values our model is supposed to reproduce. The Franke function is a weighted sum of four expontenials that have been largerly used to test fitting algorithms (9). It is defined on $x, y \in [0, 1]$ and given by

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

In addition, we will explore how adding stochastic, normally distributed ($\mathcal{N}(\iota, \infty)$) noise to the function output will affect the model performance. We will indicate in the text or figures whenever this applies.

## 3.2 Feature Matrix and Data Processing

Here we use OLS, ridge, and Lasso linear regression on a two-variable input polynomial feature matrix to reproduce the Franke function output. The polynomial is defined for $\boldsymbol{x}, \boldsymbol{y} \in [0, 1]$ as:

$$P(\boldsymbol{x}, \boldsymbol{y}) = \beta_0 + \beta_1\boldsymbol{x} + \beta_2\boldsymbol{y} + \beta_3\boldsymbol{x}^2 + \beta_4\boldsymbol{y}^2 + \beta_5\boldsymbol{x}\boldsymbol{y} + \cdots + \beta_*\boldsymbol{x}^n + \cdots + \beta_k\boldsymbol{y}^n \tag{35}$$

and the degree of this polynomial is defined as the sum of the maximum degrees of $x$ and $y$'s non-zero terms.

Moreover, we scale the input data using the Standard Scaler from the "Scikit-learn" Python library (11). Data scaling reduces the influence of possible outliers and normalizes the data. Thus, it can lead to better model performance. The scaling procedure is as follows:

$$y_i' = \frac{y_i - \bar{y}}{\sigma}, \tag{36}$$

with $\bar{y}$ being the mean of the original $y$ values and $\sigma$ being the original standard deviation. Accordingly, the mean of the scaled data $\mu = 0$ and $\sigma = 1$.

We then analyze the bias variance tradeoff using the different resampling techniques as presented in sections 2.4.1 and 2.4.2, i.e. random subsetting, bootstrapping, and k-fold cross-validation. This routine was repeated for each of the above mentioned regression methods.

## 3.3 Terrain Data

Finally, we used the same polynomial function and regression methods to model real digital terrain data from Norway (Figure 3). It was retrieved as a GeoTIFF raster file (`SRTM_data_Norway_1.tif`) from https://github.com/CompPhysics/MachineLearning [accessed: 01-10-2020]. The full raster matrix contains $n = 6,485,401$ entries where each value represents the modelled height of the terrain over mean sea level (m.a.s.l.). Due to constraints in computational power and time, we extracted a random subset of the data (70x70), therefore, $n = 4,900$. We scaled the input features ($\boldsymbol{x}$ and $\boldsymbol{y}$ values of the raster) and performed model training and evaluation using the different resampling techniques, similarly to the procedures described in the previous section. Moreover, we fit polynomials of different orders to investigate the bias-variance-tradeoff in this example.

**Figure 3:** *The location of the full terrain data set in southern Norway (black and white rectangle; brighter colors indicate higher terrain [m.a.s.l.]). Base map from https://www.openstreetmap.org.*

# 4 Results and Discussion

## 4.1 Franke Function

Figure 4 shows 3D plots of the Franke Function output with randomly added noise and the respective predictions from the OLS, Ridge and Lasso regression methods. The polynomials were chosen up to $n = 5$, and the number of data points were chosen to be $N = 70^2$. We chose to apply a magnitude equal to 0.1 to the random, normally distributed noise that we add to the Franke Function output. A larger factor would lead to a clearer distinction of bias-variance with increasing polynomial degree, but the purpose here was to avoid distorting the Franke Function too much and more closely preserve the overall shape of the Franke function as shown in Figure 4. We can see that the predictions seem to reproduce the general shape of the function correctly, but underestimate the maxima of the Franke Function. Table 1 shows the different MSE and $R^2$ scores for the different methods. Ridge and OLS performed roughly equally well, with the Ridge performing slightly better, meanwhile the Lasso regression resulted in the lowest $r^2$ and highest MSE using the chosen hyperparameter $\lambda = 10^{-4}$. The reason for this can be inferred from Figure 4 (d in contrast to a-c) - the Lasso regression leads to a more distinct regularization of the output, thereby reducing ("flattening") local maxima.

**Table 1:** *The MSE and $R^2$-scores for the different regression methods with $n = 5$ and $\lambda = 10^{-4}$ on the* ***Franke function***

| | OLS | RIDGE | LASSO |
|---|---|---|---|
| MSE | $\approx 0.01220$ | $\approx 0.01150$ | $\approx 0.01644$ |
| R2 | $\approx 0.86613$ | $\approx 0.86676$ | $\approx 0.82006$ |

Figure 8 depicts the $\boldsymbol{\beta}$ confidence intervals for both OLS and Ridge regression on the Franke function. The $\boldsymbol{\beta_{OLS}}$ is higher for OLS than for Ridge for almost all of the $\beta_i$. In addition, many of the confidence intervals for $\boldsymbol{\beta_{OLS}}$ appear larger which indicates a higher variability in the respective parameters. This outcome was expected as it can be shown that the variance of the ridge estimators is equal or lower to the variance of the respective OLS estimators. For a proof and additional details refer to (12).

The bias variance tradeoff for the OLS is shown in figure 17. With our chosen level of noise, we experienced unexpected test errors at low polynomial degrees which were lower than the train error. To combat these random misfits, an average of the train and test error over 100 iterations was computed.

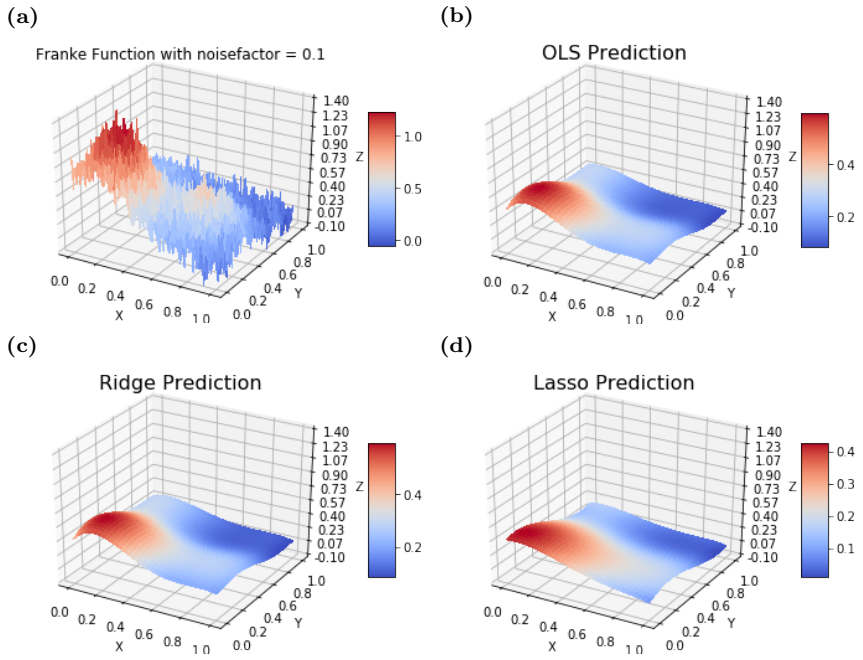



**Figure 4:** *Different regression methods on the terrain data (**a**): Surface plot of the Franke function with noisefactor* $= 0.1$. *(**b**): Surface plot of the predicted Franke function using OLS (**c**): Surface plot of the predicted Franke function using Ridge, hyperparameter* $\lambda = 0.0001$ *(**d**): Surface plot of the predicted Franke function using Lasso, hyperparameter* $\lambda = 0.0001$

The Bias-Variance tradeoff is shown in Figure 5 for the ridge regression, and in Figure 6 for Lasso regression. The ridge regression shows an increase in bias with increasing $\lambda$, and even greater increase with lower polynomial degree $n$. The lasso regression shows a very high increase in the bias with increasing $\lambda$, and the variance is completely removed after $\lambda = 1$.

For $\lambda \geq 1$, the lasso regression has a remarkably higher bias and smaller variance than that of the ridge regression. This could be explained by the way lasso regression instils more bias into the function by operating with the $l_1$ norm constraint and thereby resulting in a very biased fit after the penalty term reaches $\lambda \geq 1$ (see 2.1.5).

Computing k-fold cross validation resulted in unexpected values based on theory using a custom implementation. Thus, for the error analyses and our study of the resampling techniques, we here limit the discussions to bootstrapping results. For the ridge regression, the k-fold cross validation is depicted in Figure 7. The variance acted as one would expect, i.e. decreasing for decreasing $\lambda$ and increasing $n$. However, the results for bias and the MSE do not follow clear patterns and appear counter-intuitive.

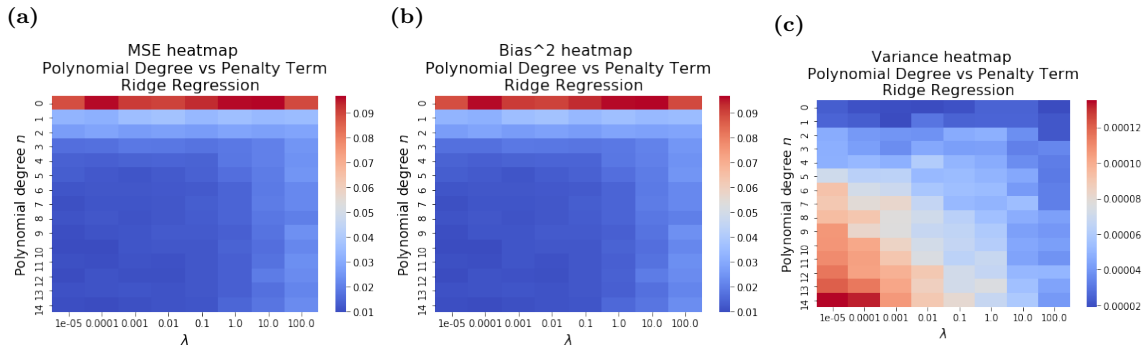



**Figure 5:** *Bias-Variance tradeoff heatmaps for Ridge Regression on the* ***Franke function*** *using bootstrapping. (**a**): MSE with increasing n and $\lambda$ (**b**): Bias$^2$ with increasing n and $\lambda$ (**c**): Variance with increasing n and $\lambda$*

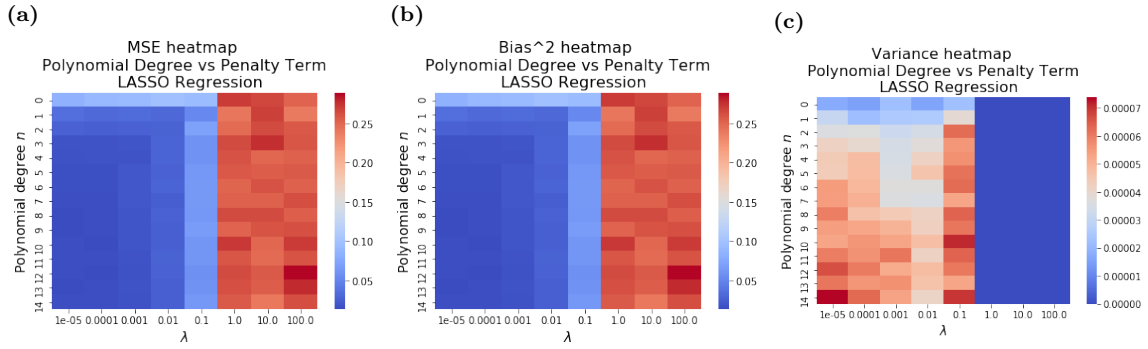



**Figure 6:** *Bias-Variance tradeoff heatmaps for Lasso Regression on the* ***Franke function*** *using bootstrapping. (**a**): MSE with increasing n and $\lambda$ (**b**): Bias$^2$ with increasing n and $\lambda$ (**c**): Variance with increasing n and $\lambda$*

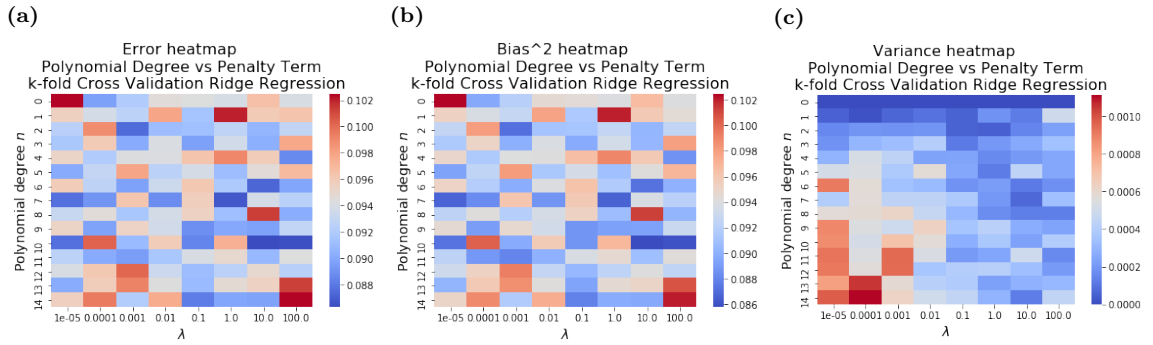



**Figure 7:** *Bias-Variance tradeoff heatmaps for Ridge Regression on the* ***Franke function*** *using k-fold cross validation.* (***a***)*: MSE with increasing* $n$ *and* $\lambda$ (***b***)*:* $Bias^2$ *with increasing* $n$ *and* $\lambda$ (***c***)*: Variance with increasing* $n$ *and* $\lambda$

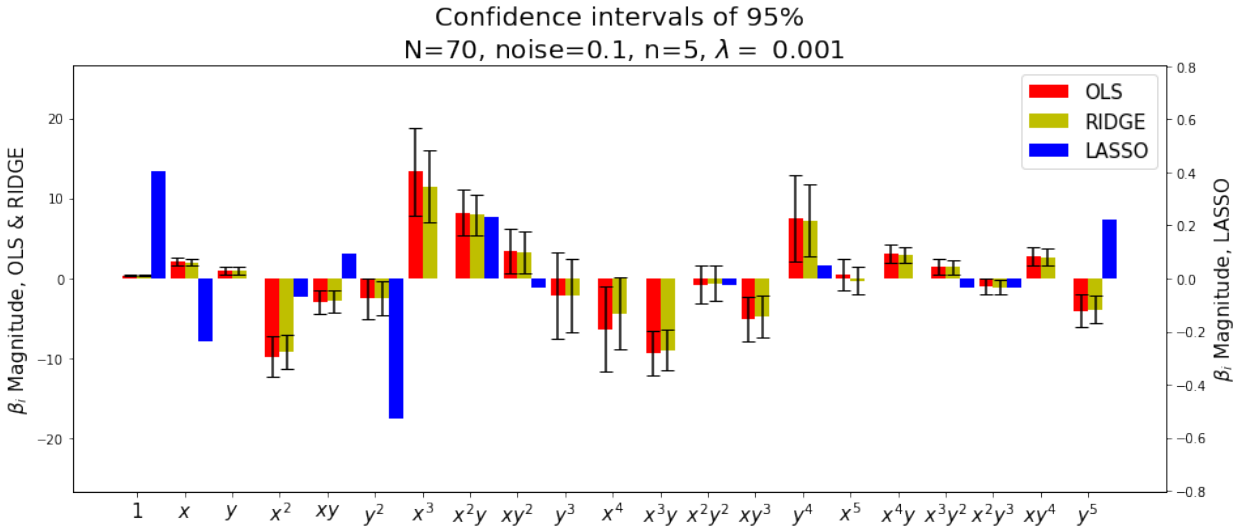



**Figure 8:** *Comparison of the* $\boldsymbol{\beta}$ *for the OLS , Ridge and the Lasso method on the* ***Franke function*** *with 95 % confidence intervals for the OLS and Ridge. Note that the scale for the Lasso coefficients is on the left hand side.*

## 4.2 Terrain Data

The terrain data was chosen as a subset of the data as stated earlier, where we reduced the resolution for a quicker computation time, see figure 9. We did not add any noise to the terrain data unlike the Franke function.

The scores of a train test split fit on the terrain data with the various regression methods is shown in table 2, and the figures in 10. The OLS performed best with the Ridge and the LASSO performing increasingly worse respectively. For this fit we chose n=10 and = $10^{-4}$. The choice of the parameters are more extensively explained in 4.2.2.

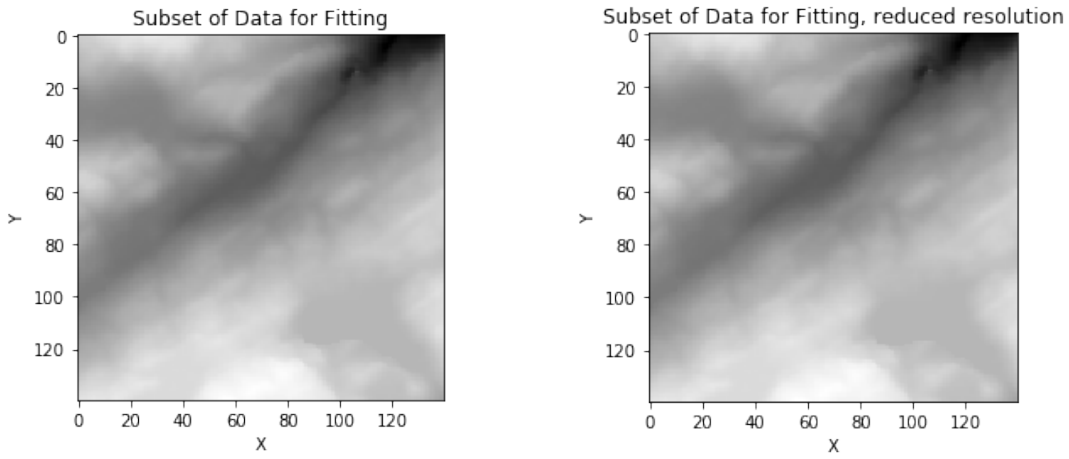
The confidence intervals of the ridge and lasso regression can be found in figure 12, and their corresponding coefficients can be found in figure 11. We can see that the Lasso and Ridge regression are severely smaller than the OLS coefficients, and the Lasso regression are only non-zero for a few $\beta_i$. This highlights the improvement adding a penalty term to our model as the fit is only slightly reduced but fewer coefficients are needed. The fit is also heavily influence by the smoothness of the data, and from figure 10 we can see that the data we are fitting our model upon is have a very distinct trend with Z increasing in the positive Y direction of the surface plot.

### 4.2.1 Linear Regressions on Random Data Split

The terrain data and the regression predictions are shown in Figure 10. As in the Franke Function, the predictions struggle with capturing the maximum values correctly.

### 4.2.2 Bias-Variance tradeoff and resampling

The bias variance tradeoff was computed using equation 30, but since $f_i$ usually is not known like our Franke function, here we used $y_i = f_i + \epsilon_i$. In figure 17 and 18 we computed the MSE, bias and variance with the OLS, using bootstrap as our resampling technique. For ridge and lasso regression, we plotted the MSE, bias and variance as heatmaps with increasing $n$ and $\lambda$, see figure 5 and 6 for the Franke function and figure 13 and 14 for the terrain data. The Bias on the terrain data converges fully towards zero, in contrast to our fits on the Franke function. This difference may be due to the noise level added to our franke function.




**(a)** **(b)**

**Figure 9:** *The subset of the terrain data we chose to model (a) and the the same area after reducing resolution for enhanced computational time (b).*

The bias-variance trade off for the ridge regression on the franke function can be found in figure 5, and for the lasso regression in figure 6. The ridge regression shows slightly increasing bias with increasing $\lambda$ and a heavy increase in bias approaching low polynomial degree. The variance is also acting expectedly, increasing with low and high $n$. We can see that our choice of fitting the franke function with $n = 5$ fits well with being in a low bias and low variance. The lasso regression increases more heavily with increasing $\lambda$, and after $\lambda \geq 1$ the variance is virtually zero and the bias is capped at a fixed amount. Our choice of predicting the Franke function with $n = 5$ and $\lambda = 0.0001$ fits well with the low-bias, low-variance zone.

The bias variance for the ridge regression on the terrain data can be found in Figure 13, and for the lasso regression in Figure 14. The ridge regression in the terrain data shows little increase of bias with increasing $\lambda$. This could be explained by the fact that the subset of the terrain data chosen fitted well with a planar solution, and the penalty would need to get needlessly large to show the bias increase heavily. The lasso regression on the terrain data however was very sensitive after $\lambda = 1$. This may be due to the $l_1$ norm constraint, picking out fewer coefficients and thereby inducing more bias into the model (see 2.1.5). However, when inspecting the amount of parameters chosen by the lasso compared with the ridge on the terrain, shown in figure 15, we see that the Lasso regression however has fewer and fewer values as a function of both increasing polynomial degree and $\lambda$. Thus the lasso regression returns almost no active parameters for $\lambda \geq 1$, thereby fitting very poorly to our data. In figure 15 we can see also see that the ridge coefficient has all its possible active parameters, $|\beta_i| > 0$, with high polynomial degree. This fits well with the theory of lasso which was explained in 2.1.5.

**Table 2:** *The MSE and $R^2$-scores for the different regression methods on the terrain data with $n = 10$ and $\lambda = 10^{-4}$.*

| | OLS | RIDGE | LASSO |
|---|---|---|---|
| MSE | $\approx 0.00694$ | $\approx 0.01889$ | $\approx 0.04756$ |
| R2 | $\approx 0.99314$ | $\approx 0.98107$ | $\approx 0.95056$ |

It is also evident that from the MSE and $R^2$ scores in table 2 that the Lasso regression performed worst among the regression methods in this example.

#### 4.2.3 General discussion

In this chapter, we have demonstrated that linear regression on x and y coordinates using a two-variable polynomial function feature input can lead to good model fits that, accordingly, reproduce real terrain data well. However, these results are a good motivation to reflect on the applicability of machine learning methods. One common goal in machine learning is to make predictions beyond the available input variable space. The models from our example of predicting terrain height based on coordinates can clearly not be used to make predictions for locations outside the subset. We know that geomorphology involves complex processes such as plate tectonics, erosion, etc., and thus the coordinates can only in some instances act as proxies for underlying "true" explanatory variables. The reason why we can still get acceptable models on the training set in our examples are spacial auto-correlations - through the resampling methods we chose, there is a high chance that the terrain height values in close proximity are similar, thus, the models will still perform acceptable on randomly left out samples. Accordingly, the methods might still have some practical use, e.g. if we were to impute missing values of the DEM. To conclude, it is always important to be aware of possible model limitations, to chose an appropriate model, and to critically evaluate predictors for a given task.

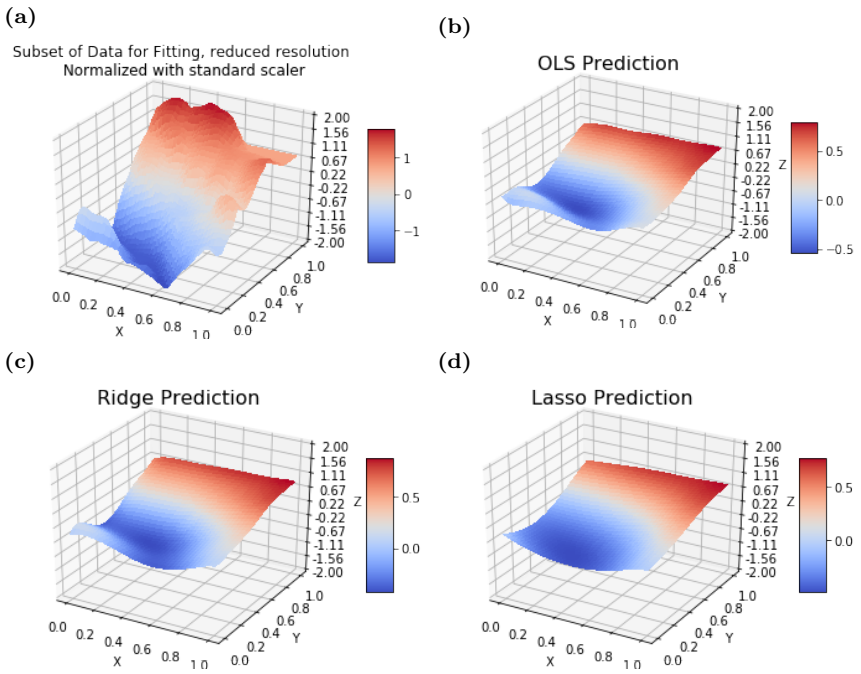



**Figure 10:** *Different regression methods on the terrain data (**a**): The surface plot for the actual data (**b**): OLS prediction of the terrain, $n = 10$ (**c**): Ridge prediction of the terrain, $n = 10$, $\lambda = 0.001$ (**d**): lasso prediction of the terrain, $n = 10$, $\lambda = 0.001$*

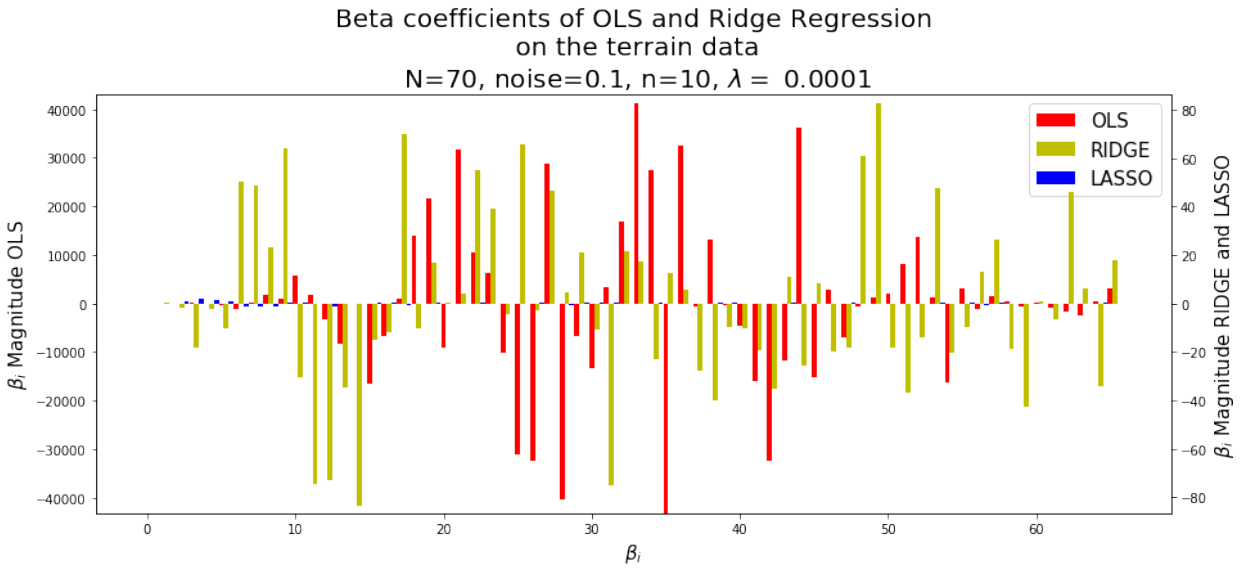


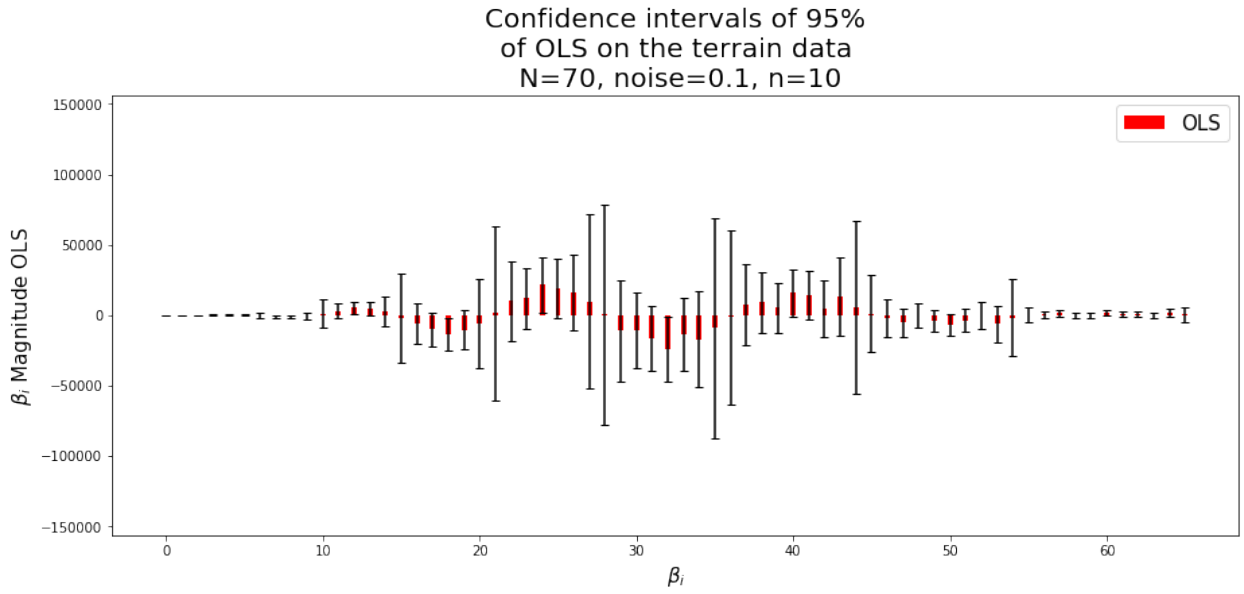

**Figure 11:** *Comparison of the* $\boldsymbol{\beta}$ *for the OLS, Ridge and Lasso method on the* ***Terrain Data****. Note that the OLS has y-scale on the left, and the Ridge and the Lasso has scale on the right.*

# 5 Conclusions and Perspectives

The methods studied in this report are widely used in supervised learning and data analysis. We have studied the Ridge and Lasso regression and their dependence on model complexity, in our case polynomial degree, and the hyperparameter $\lambda$. The Ridge regression and OLS resulted in similar model performance, where Ridge regressions provided slightly better scores for the Franke function, while the OLS performed slightly better for the real terrain data. However $\lambda$ and the polynomial degree $n$ should be finely tuned for fitting the data, and if not taking proper precautions, may lead to a very uninformative or badly performing model. The lasso regression is good at estimating the most important features for the model as it required the least features to produce a sufficient estimate.

Regrettably, we deem the results from the k-fold cross validation to not be usable as they gave unrealistic values, and we therefore limited ourselves to using bootstrap as our resampling technique. Based on the theoretical background, we would expect both methods to provide similar overall patterns concerning the Bias-Variance tradeoff and robust results, and both are widely used in current machine learning literature.

**(a)**




**(b)**

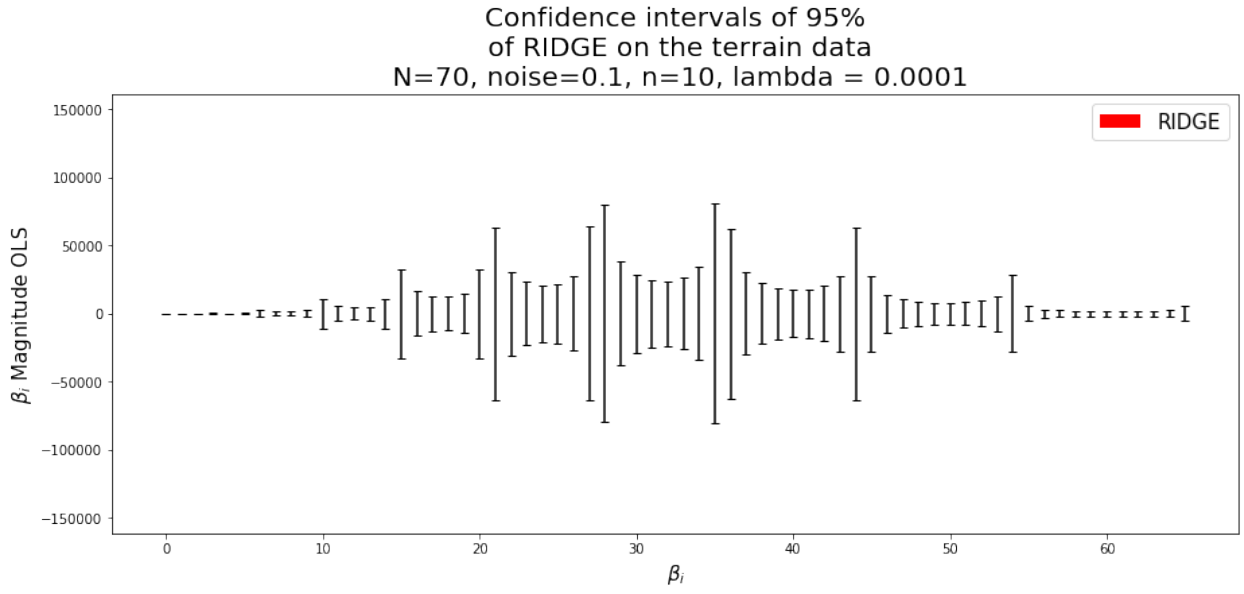



**Figure 12:** *Comparison between the 95% confidence intervals for OLS and Ridge on the terrain data (**a**): OLS confidence intervals on the terrain data (**b**): Ridge confidence intervals on the terrain data*

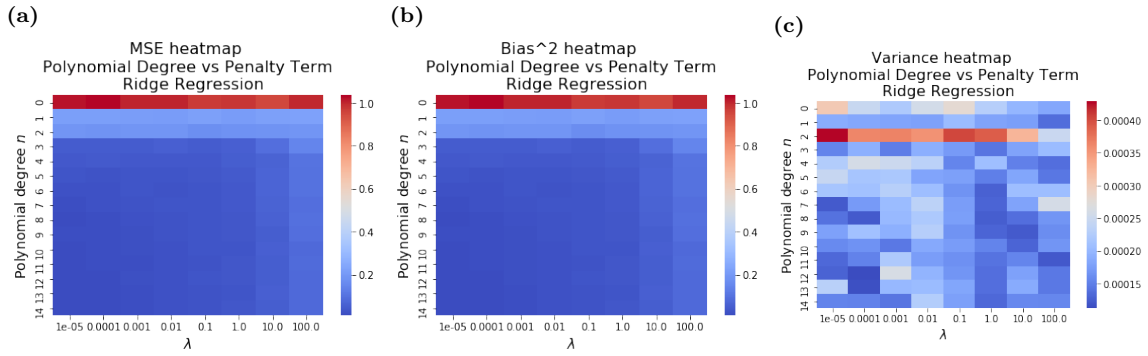



**Figure 13:** *Bias-Variance tradeoff heatmaps for Ridge Regression on the* ***Terrain Data*** *using bootstrapping.* (***a***)*: MSE with increasing n and* $\lambda$ (***b***)*: Bias*$^2$ *with increasing n and* $\lambda$ (***c***)*: Variance with increasing n and* $\lambda$

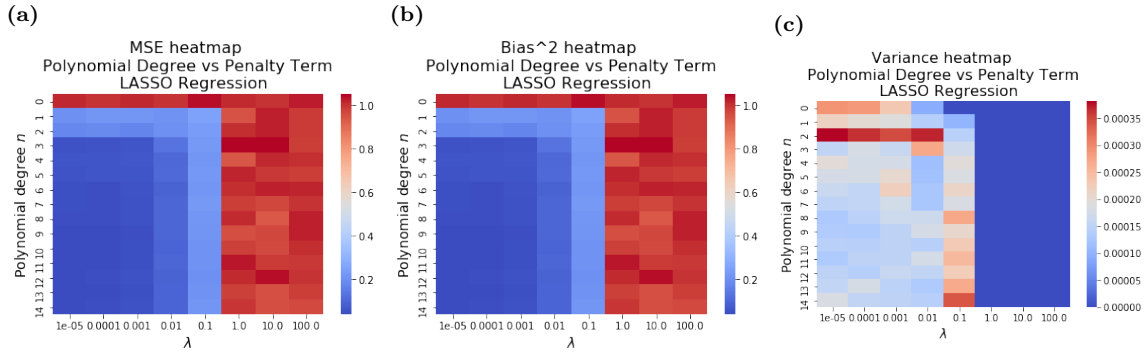



**Figure 14:** *Bias-Variance tradeoff heatmaps for Lasso Regression on the* ***Terrain Data*** *using bootstrapping.* (***a***)*: MSE with increasing n and* $\lambda$ (***b***)*: Bias*$^2$ *with increasing n and* $\lambda$ (***c***)*: Variance with increasing n and* $\lambda$

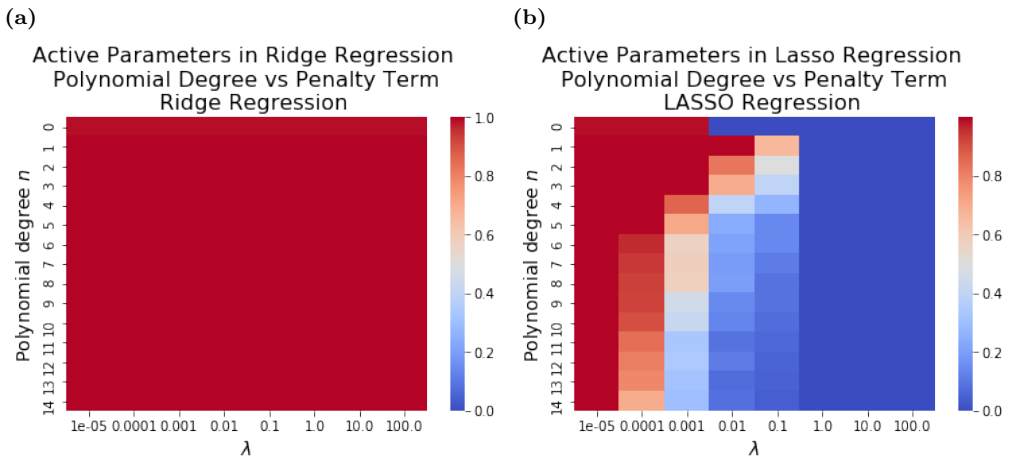



**Figure 15:** *Heatmap of number of $\beta_i > 0$ for $\lambda$ and $n$ on the terrain data (**a**): Ridge (**b**): LASSO*

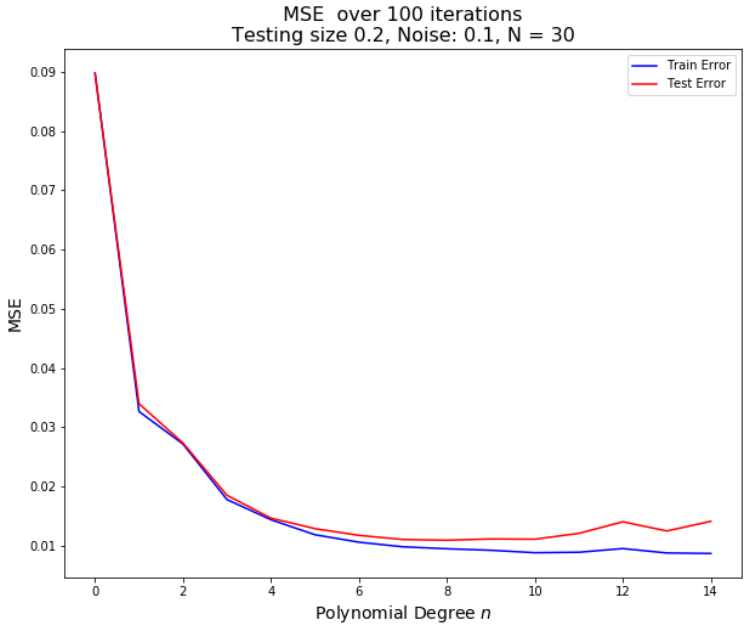



**Figure 16:** *Expected MSE error, averaged over 100 iterations.*

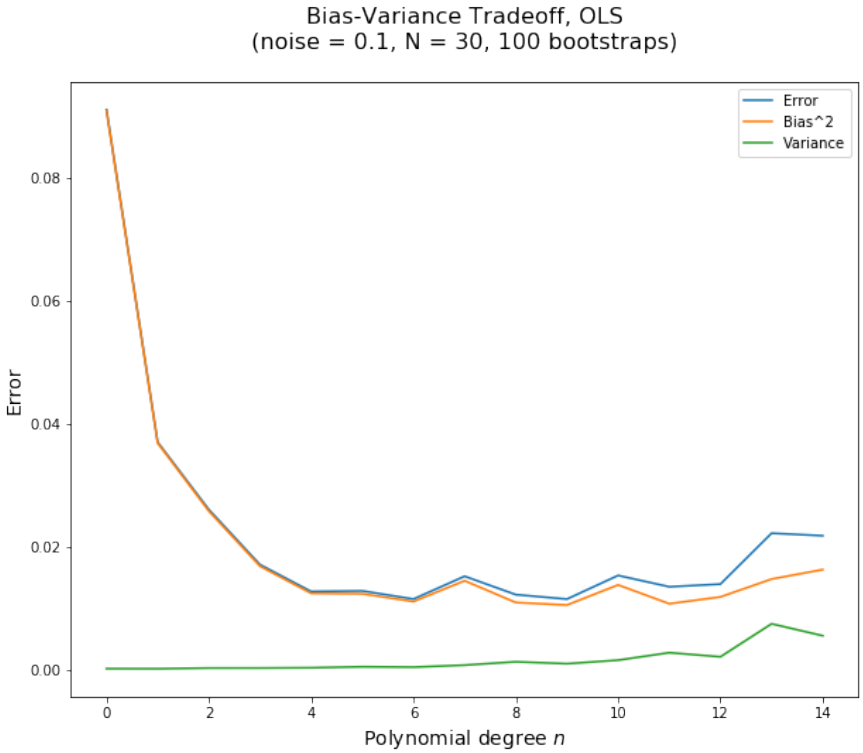



**Figure 17:** *Bias-Variance Tradeoff shown with OLS regression with bootstrapping as resampling technique on the* ***Franke function****.*

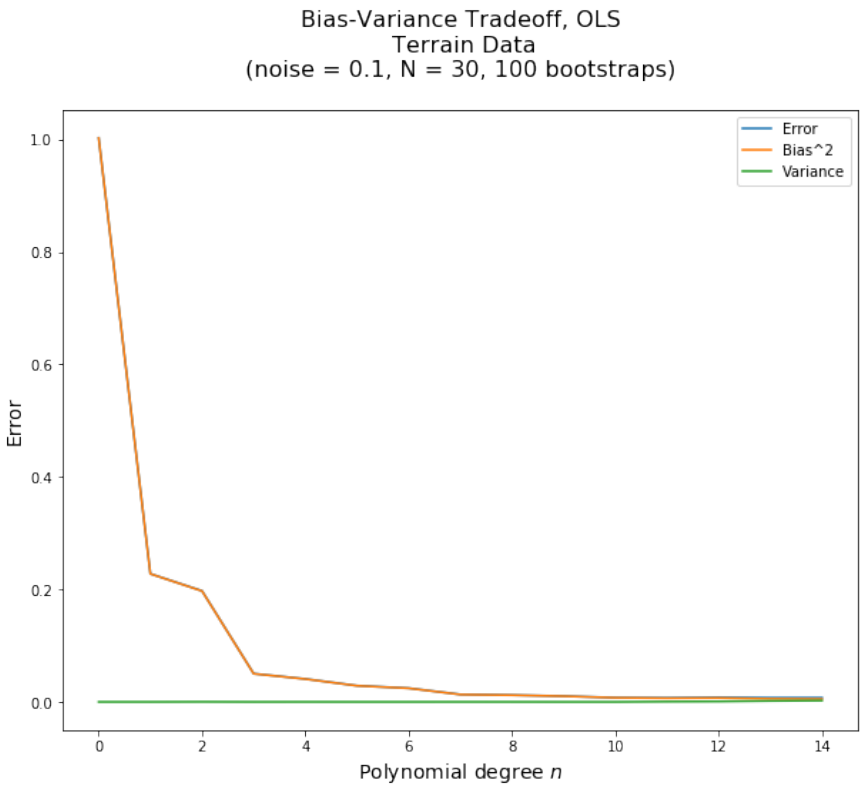



**Figure 18:** *Bias-Variance Tradeoff shown with OLS regression with bootstrapping as resampling technique on the* ***terrain data****.*

The methods employed can be improved upon further study and show great potential to improve linear regression fits.

# 6 Documentation

The figures and python scripts can be found in the github of one of the group members:
https://github.com/hakonbol/FYS-STK4155_AUT2020/tree/master/Project%201
The main script can be found at:
https://github.com/hakonbol/FYS-STK4155_AUT2020/blob/master/Project%201/Project%201%20main.ipynb
The function library can be found at:
https://github.com/hakonbol/FYS-STK4155_AUT2020/blob/master/Project%201/functions.py
The test run for benchmarking:
https://github.com/hakonbol/FYS-STK4155_AUT2020/blob/master/Project%201/Project%201%20Test%20Runs.ipynb
And the figures can be found at:
https://github.com/hakonbol/FYS-STK4155_AUT2020/tree/master/Project%201/Results

# References


[SVD] Singular value decomposition (svd). https://www.cse.unr.edu/~bebis/CS791E/Notes/SVD.pdf. Accessed: 2020-10-10.

[2] Aragay, V. R. (2018). Boston housing. https://github.com/rromanss23/Machine_Leaning_Engineer_Udacity_NanoDegree/tree/master/projects/boston_housing. Accessed: 2020-10-10.

[3] Chen, M., Hao, Y., Hwang, K., Wang, L., and Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. *Ieee Access*, 5:8869–8879.

[4] Fatima, M., Pasha, M., et al. (2017). Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, 9(01):1.

[5] Hastie, T., Tibshirani, R., and Friedman, J. (2017). The elements of statistical learning the elements of statistical learning, vol. 27.

[6] Hjorth-Jensen, M. (2020a). Lecture notes: Week34: Introduction to applied data analysis and machine learning. https://compphysics.github.io/MachineLearning/doc/pub/week34/html/week34.html.

[7] Hjorth-Jensen, M. (2020b). Lecture notes: Week35: Linear regression and review of statistical analysis and probability theory. https://compphysics.github.io/MachineLearning/doc/pub/week35/html/week35.html.

[8] Hjorth-Jensen, M. (2020c). Lecture notes: Week36: Resampling techniques and ordinary least square. https://compphysics.github.io/MachineLearning/doc/pub/week36/html/week36.html.

[9] Hjorth-Jensen, M. (2020d). Project 1 on machine learning. https://compphysics.github.io/MachineLearning/doc/Projects/2020/Project1/pdf/Project1.pdf.

[10] LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2. Accessed: 2020-10-10.

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

[12] Taboga, M. (2017). Ridge regression. Lectures on probability theory and mathematical statistics, Third edition. Kindle Direct Publishing. Online appendix. https://www.statlect.com/fundamentals-of-statistics/ridge-regression [accessed 08-10-2020].