Regression analysis on terrain data

Mathias M. Vege

October 10, 2018

Abstract

An often used tool in Machine Learning is that of Linear Regression and resampling. For this paper, we investigated how to use Ordinary Least Squares, Ridge and Lasso regression together with resampling techniques such as bootstrapping, k-fold Cross Validation and Monte Carlo Cross Validation. After first studying the Franke function and using that as a proving ground for the different models, we moved on to study terrain data from a region outside Stavanger. The optimal model for this region turned out to be OLS with $R^2 = 0.91$ and MSE$= 1.26 \cdot 10^{-3}$, which is an acceptable result considering the scope of this report.

Contents

1 Introduction

A now-fully emergent field of data analysis, is that of regression and resampling. Included as a subset of the field of machine learning, regression is widely used as tool of prediction. Together with resampling we are offered ways of estimating the error of our models.

One common way to investigate the efficacy of a model is to use the Franke function[3]. This function has complexities which makes it suitable to use as a testing ground for machine learning. After studying the noise deependency on the quality of fit R^2, we moved on to investigating how the data fitting would behave for real terrain data.

We will start of by introducing the different regression methods, then resampling techiniques before giving a brief description of the implementation. After that we will present the results in an orderly fashion, before delving into the discussion and conclusion of how we can choose the optimum fit parameters for the different regression methods and how well the differet methods reproduce the data. Whitout further a due, let us start of by introducing some basic notation on linear regression.

2 Methods and theory

Let us start of by summing up notational conventions. We will denote a *case* or *sample* by x_i, where $i = 0, \cdots, n$. This will have a corresponding *response* or *outcome* $y_i = f(x_i)$. We typically want to estimate f by some approximate function $\tilde{f}$. As we shall see, one method is by performing linear regression.

2.1 Linear regression

As mentioned, linear regression is a method of performing a fit of parameters x_i to a given data set y_i. From this we wish to minimize the error between the *response* or *real* data points y_i, and some approximation $\tilde{y}_i = \tilde{f}(x_i)$. This defines our loss function, which we will seek to minimize. In linear regression this will be by building an *design matrix* built from p *explanatory variables* or *predictors* $\boldsymbol{x_i} = x_i^0 + x_i^1 + \cdots + x_i^p$. The explanatory variables is simply the degrees of the polynomial we seek to approximate f by. So far, we have not mentioned the variables which will be tuned to the model, the *regression parameters* $\beta = (\beta_1, \ldots, \beta_p)^T$. The keen-eyed reader will now notice that the predictors and regression parameters all are of equal number, and may have guessed that they form our linear approximation - this is true:

$$\tilde{y}_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$

From this, we get a neat matrix equation,

$$\boldsymbol{y} = \beta \boldsymbol{X} + \boldsymbol{\epsilon} \tag{1}$$

We see that our goal has immediately become to minimize β. The definition of the error vector $\boldsymbol{\epsilon}$ will be given later.

The simplest case of such a linear regression minimization problem is just a simple line [5, ch. 3.1, p. 61]. However, this can be generalized greatly and we end up with the standard go-to method called *Ordinary Least Squares.*

2.1.1 Ordinary Least Squares regression(OLS)

To begin our journey towards a solution of OLS, we start by defining the *loss function* for OLS.

$$\min_{\beta\in\mathbb{R}^p} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 = \min_{\beta\in\mathbb{R}^p} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T\beta - y_i\right)^2 \tag{2}$$

We see that the goal is to minimize the L$_2$ norm between the response y_i and the fitted predictors $\boldsymbol{x}_i^T\beta$ [6, ch. 4, p. 21].

To find a solution to the loss function, we can start by observing the dimensions of equation (1),

$$\underbrace{\boldsymbol{y}}_{\mathbb{R}^n} = \underbrace{\boldsymbol{X}}_{\mathbb{R}^{n\times p}}\underbrace{\beta}_{\mathbb{R}^p} \tag{3}$$

We then see that in order to free β, we must multiply by the transposed of $\boldsymbol{X}$ such that we can take the inverse and solve for β.

But, before we can do that we need to define what the optimal β vector is. We start by defining a function $Q(\beta)$ which gives us the squared error of the spread,

$$\begin{aligned} Q(\beta) &= \sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \\ &= (\boldsymbol{y} - \tilde{\boldsymbol{y}})^T(\boldsymbol{y} - \tilde{\boldsymbol{y}}) \end{aligned}$$

which is the matrix equation

$$Q(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta) \tag{4}$$

This equation is called the cost function [6, p. 12]. Before moving on, we should not that we can generalize it to

$$Q(\boldsymbol{y}, g(\boldsymbol{x})) = \sum_i (y_i - g(\boldsymbol{x})) \tag{5}$$

where $g(\boldsymbol{x})$ is some function for the predictor. In OLS, this is $\beta_i\boldsymbol{x}_i$.

The minimum of the cost function (4) can be found by differentiating by β and setting the result equal to zero.

$$\begin{aligned} \frac{\partial Q(\beta)}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j}\sum_{i=0}^{n-1}(y_i - \beta_0 x_{i0} - \beta_1 x_{i1} - \ldots \beta_{p-1}x_{i,p-1})^2 \\ &= -2\sum_{i=0}^{n-1} x_{ij}(y_i - \beta_0 x_{i0} - \beta_1 x_{i1} - \ldots \beta_{p-1}x_{i,p-1}) \end{aligned}$$

where the element x_{ij} is picked out by the differentiation w.r.t. β_j. Performing all of the derivatives leaves us with the matrix equation,

$$\frac{\partial Q(\beta)}{\partial \beta} = X^T(\boldsymbol{y} - \boldsymbol{X}\beta) \tag{6}$$

which we require to be zero in order to find an optimal β. From this the *residues* or *error vector* $\boldsymbol{\epsilon}$ is defined,

$$\boldsymbol{\epsilon} = \boldsymbol{y} - \tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\beta \tag{7}$$

The residues is the error between the *approximation* and the *real* data, and is not to be confused with the inherent error in the data. The solution should now be obvious. We need to solve equation (6) for $Q(\beta) = 0$,

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) = 0.$$

As alluded to in (3), we now need to multiply by the inverse of $\boldsymbol{X}^T\boldsymbol{X}$, which yields after splitting the parenthesis

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

And thus, the β-values is simply

$$\beta = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left(\boldsymbol{X}^T\boldsymbol{X}\right)\boldsymbol{y} \tag{8}$$

This solution does not account for errors in the measurements or responses. Say there is an σ_i associated with each measured y_i, we will have to use the χ^2 function as a our function to minimize [4, see notes on regression]. We will have to rescale our solution by $\boldsymbol{X} \to \boldsymbol{A} = \boldsymbol{X}/\boldsymbol{\Sigma}$ where Σ is a diagonal matrix of every uncertainty σ_i.

2.1.2 Ridge regression

The idea of Ridge regression is to add a small term λ to constraint,

$$\beta(\lambda) = \min_{\beta\in\mathbb{R}^p}(||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda||\beta||_2^2) \tag{9}$$

This amounts to finding a solution in parameters space β which tangents a circle defined by the L^2 norm of the constraint (9).

As shown in [8], the solution to the Ridge regression, is to replace the inverse in the OLS solution for β (8), by

$$\boldsymbol{X}^T\boldsymbol{X} \to \boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I} \tag{10}$$

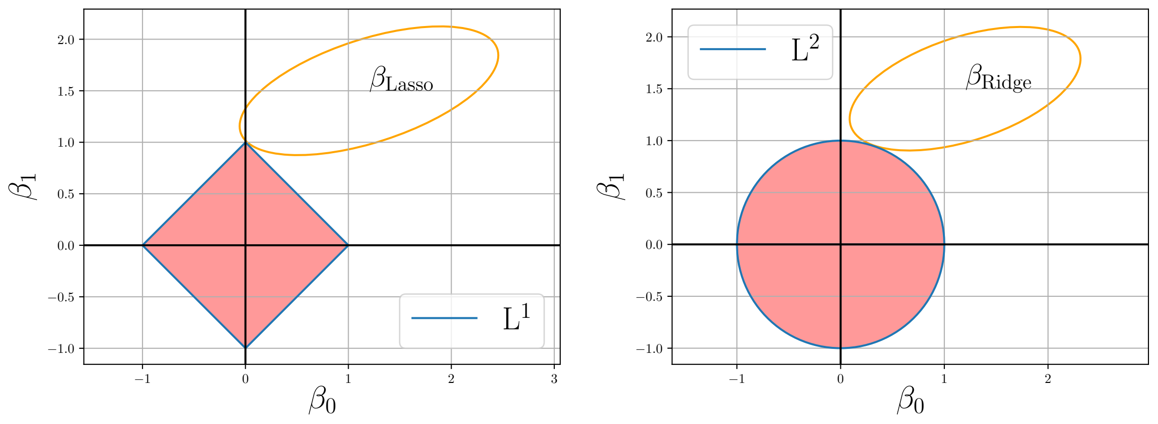
with $\boldsymbol{I}$ as the identity matrix.

The idea of adding an $\lambda\boldsymbol{I}$ can aid in resolving issues surrounding singularities in the inverse which can be troublesome when dealing with a large parameter space β[4, see notes on regression, p. 17].

2.1.3 Lasso regression

The idea behind the Lasso regression is similar to that of Ridge regression. However, it comes with a few of its own benefits, such that it tends to zero out β coefficients. To see this, we can take a look at the constraint function,

$$\beta(\lambda) = \min_{\beta\in\mathbb{R}^p}(||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda||\beta||_1^2) \tag{11}$$



(a) L^1 norm for Lasso.

(b) L^2 norm for Ridge.

Figure 1: In the figure 1b we see how the search space for the Ridge regression, where we define a success when the β parameter hits the circle. The search space for Lasso is viewed on the figure to the right 1a.

where we are taking the L^1 norm(popularly called Taxicab or Manhattan norm) in the λ term, $\lambda||\beta||_1^2$. Since the L^1 norm is defined as

$$||\boldsymbol{a}||_1 = |a_0| + \cdots + |a_{n-1}| = \sum_{i=0}^{n-1} |a_i|$$

To visualize and better understand the difference between Ridge and Lasso regression, we can take a look at figure 1. Here we see the difference to be that the Ridge will tend to get more β_i values as zero, due to the L^1 norm having a greater chance of *"hitting"* the corners of the diamond in figure 1a.

2.2 The Bias-Variance decomposition

It is possible to define variance of the predicted values $\tilde{y}$. If we take the estimator of the general cost function for linear regression (5),

$$E[Q(g(\boldsymbol{x}))] = E\left[\sum_i (y_i - g(\boldsymbol{x_i}))^2\right]$$

it can be shown that we can decompose this into the the variance of $\boldsymbol{y}$, bias and noise[6, 4].

$$\text{MSE} = \text{Bias}^2 + \text{Var} + \text{Noise} \tag{12}$$

The bias is defined as

$$\text{Bias}^2 = \sum_i (\tilde{y}_i - E[\boldsymbol{y}])^2, \tag{13}$$

which is measure of how much the prediction deviates from the real data.

A note on the Noise is needed here before we move on. The noise is defined as the the variance in the real data, Noise $= \mathrm{Var}(\epsilon_{\mathrm{noise}})$. Since we cannot know the noise beforehand, this will be gobbled up into the bias, as that is a measure of the deviation in real data and prediction. Since the noise is irreducible in the sense we cannot remove the initial noise in the data, the MSE will never lie below $\mathrm{Var}(\epsilon)$ [5, ch. 2, p. 34].

The variance in (12) is defined as the variance in in the prediction, $\tilde{\boldsymbol{y}}$.

The MSE in (12) is the Mean Square Error is defined as,

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2, \tag{14}$$

and is simply the averaged squared errors.

2.3 R^2 score

A assessment of the fit quality is given by the R^2 score,

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{\boldsymbol{y}})^2} \tag{15}$$

with $\bar{\boldsymbol{y}}$ defined as

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

2.4 Determining the best fitting procedure

An ad-hoc method of determining the best fitting procedure will be used by looking at,

$$\mathrm{Score} = \sqrt{(1 - R^2)^2 + (\mathrm{MSE})^2} \tag{16}$$

2.5 Bootstrapping

Bootstrapping is a resampling technique particularly useful for small datasets or observables with difficult-to-asses statistics[2]. The bootstrap technique is inspired by the Jackknife method[1], but differs since we randomly selects a n new samples with replacement of the original dataset $\{y_i\}$ a total of N_{bs} times. We use the N_{bs} samples to find the MSE, variance and bias.

A basic for bootstrap algorithm can be set up as following,

Algorithm 1 Bootstrap

```
1: Split the dataset $\boldsymbol{X}_\mathcal{L} = \{y_i, \boldsymbol{x}_i\}$ into a training set and a test set(or holdout set), $\boldsymbol{X}_{\mathcal{L},\text{train}}$ and $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
2: for $N_\text{bs}$ times do
3:     Build a new data set $\boldsymbol{X}_{\mathcal{L},\text{bs}} = \{y_{i,\text{bs}}, \boldsymbol{x}_{i,\text{bs}}\}$ with $n$ randomly drawn with replacement values, from the training set $\boldsymbol{X}_{\mathcal{L},\text{train}}$.
4:     Fit the bootstrapped dataset to a given model.
5:     Evaluate the model on on the test set $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
6:     Store relevant results.
7: end for
8: Perform final statistics on either the collection of bootstrapped datasets and/or take the averages on the stored results. Depending on the type of variable you are sampling, you might need to use latter one.
```

2.6 k-fold Cross Validation

The Cross Validation(CV) is a resampling technique which is similar in thought to the Jackknife method, but differs at a few critical places. Details on the k-fold CV method can be found in the book An Introduction to Statistical learning by James et al. [5, ch. 5.1], and can be summarized as following,

Algorithm 2 k-fold Cross Validation

```
1: Shuffle the dataset $\boldsymbol{X}_\mathcal{L} = \{y_i, \boldsymbol{x}_i\}$.
2: Split the data set into training and test data, $\boldsymbol{X}_{\mathcal{L},\text{train}}$ and $\boldsymbol{X}_{\mathcal{L},\text{test}}$.
3: Split the training data into $k$ folds. $\boldsymbol{X}_{\mathcal{L},\text{train}} \rightarrow \{\boldsymbol{X}_{\mathcal{L},i_k}\}$
4: for each $i_k$ in the $k$ folds do
5:     Select the sets $\tilde{\boldsymbol{X}}_\mathcal{L} = \boldsymbol{X}_\mathcal{L}/\{\boldsymbol{X}_{\mathcal{L},i_k}\}$ as training sets, excluding the $k$'th set which is used a holdout set. Note: in order to evaluate the Bias-Variance score (12), the model will not be compared on the $i_k$ test set.
6:     Fit the $\tilde{\boldsymbol{X}}_\mathcal{L}$ to the model which is being used.
7:     Evaluate the $\tilde{\boldsymbol{X}}_\mathcal{L}$ on the test set $\boldsymbol{X}_{\mathcal{L},\text{test}}$, and perform necessary statistics.
8: end for
9: Summarize the evaluations and statistics gained from the $k$ folds.
```

The Cross Validation comes in many different flavors. Among popular ones are the Monte-Carlo Cross Validation(MCCV). The idea is the same, except that we randomly select the i_k'th set to leave out and run this N_{MC} times instead of just k times. The samples selected each time are selected without replacement, but one can have the same samples over many Monte Carlo iterations. E.g. the indices of the first set can be $12, 32, 56, \mathbf{42}, 94, 83$, and $49, 2, 59, \mathbf{42}, 51, 73$, where $\mathbf{42}$ repeats.

3 Implementation

The full implementation can be found at [9]. The OLS and Ridge regressions have been implemented, as well as the resampling methods Bootstrap, k-fold CV and MCCV. When

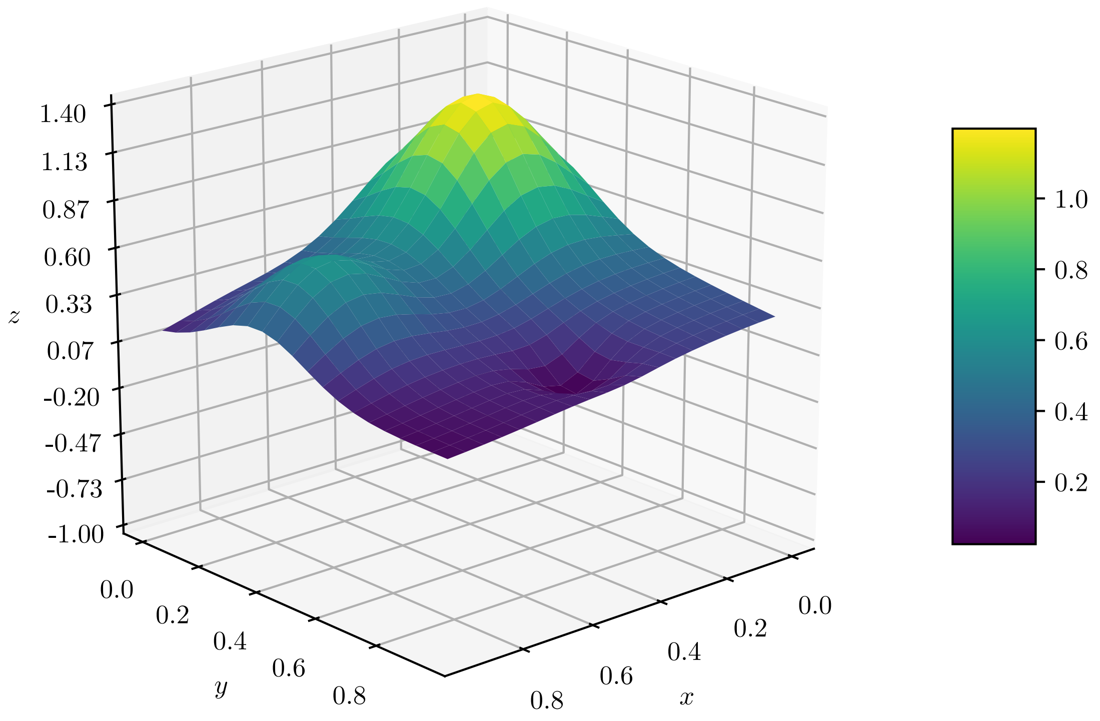


Figure 2: The mapping of $x = [0, 1]$, $y = [0, 1]$ through the Franke function (20).

taking the inverse in OLS regression (8) and the inverse in Ridge regression (10) in order to find the β coefficients a Singular Value Decomposition were used.

Using scikit-learn[7] methods for OLS, Ridge and Lasso regression were implemented, as well as k-fold cross validation.

4 Results

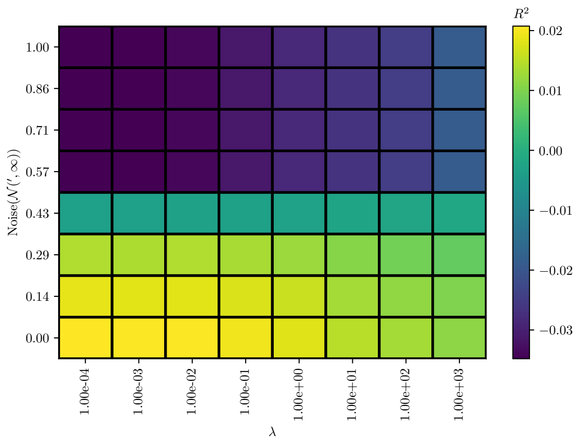
The results presented are split in two section: the results governing method verification using the Franke function(20) and methods modeling terrain data as retrieved from `https://earthexplorer.usgs.gov`

4.1 Verification through the Franke function

The Franke function $F(x, y)$(20) maps coordinates x, y to a z-coordinate. Such a mapping for points $x = [0, 1]$, $y = [0, 1]$ can be viewed in figure 2.

In figure 3, we can observe the R^2-score and MSE and how Lasso regression behaves for different levels of noise and λ-parameters. The same can be seen for Lasso in figure 4.

The bias-variance for the Franke function can be seen in figure 5.


(a) A plot of R^2 versus noise and λ

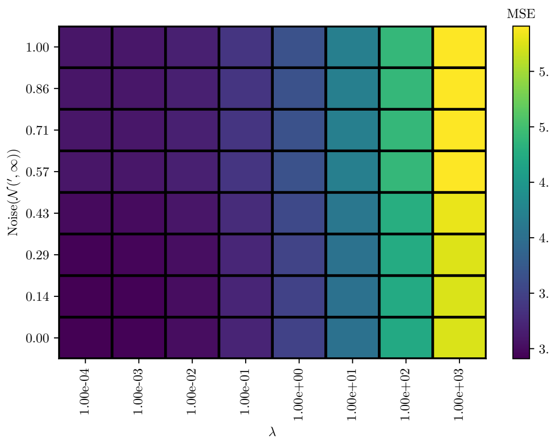
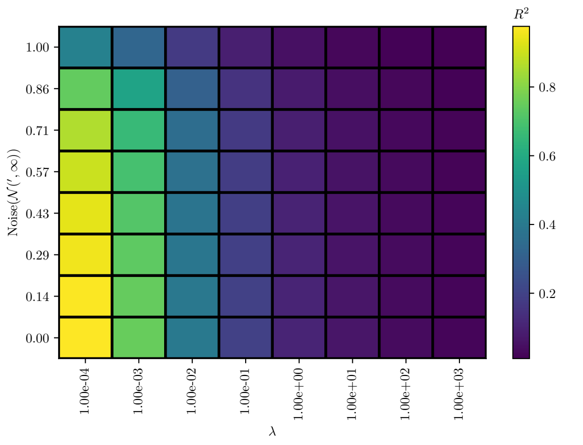

(b) A plot of MSE versus noise and λ

Figure 3: A plot of how MSE and R^2 varies as function of noise and λ with Lasso regression.


(a) A plot of R^2 versus noise and λ

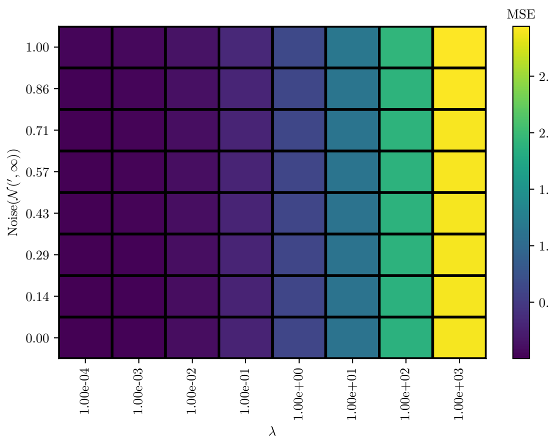
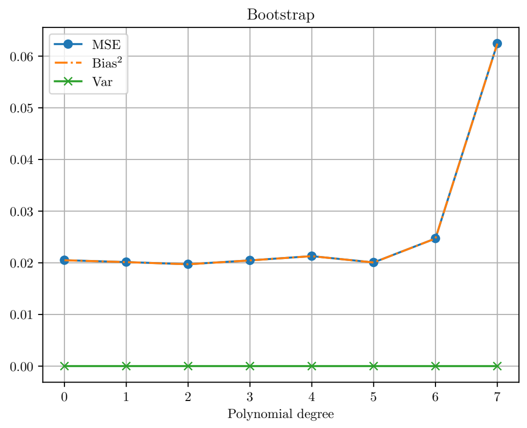
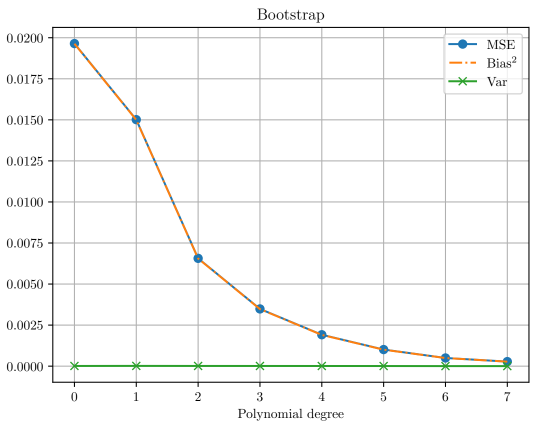

(b) A plot of MSE versus noise and λ

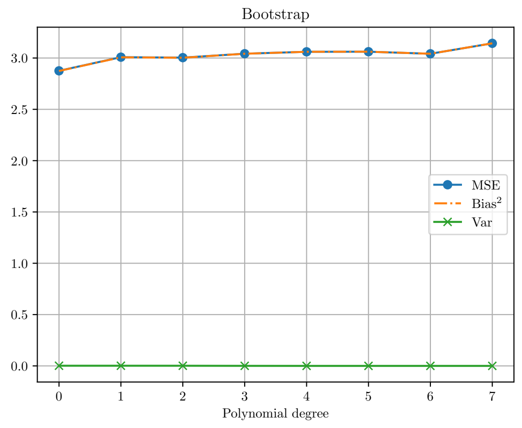
Figure 4: A plot of how MSE and R^2 varies as function of noise and λ with Ridge regression.



(a) Ridge



(b) OLS



(c) Lasso

Figure 5: Variance-bias plots of the OLS, Ridge and Lasso.

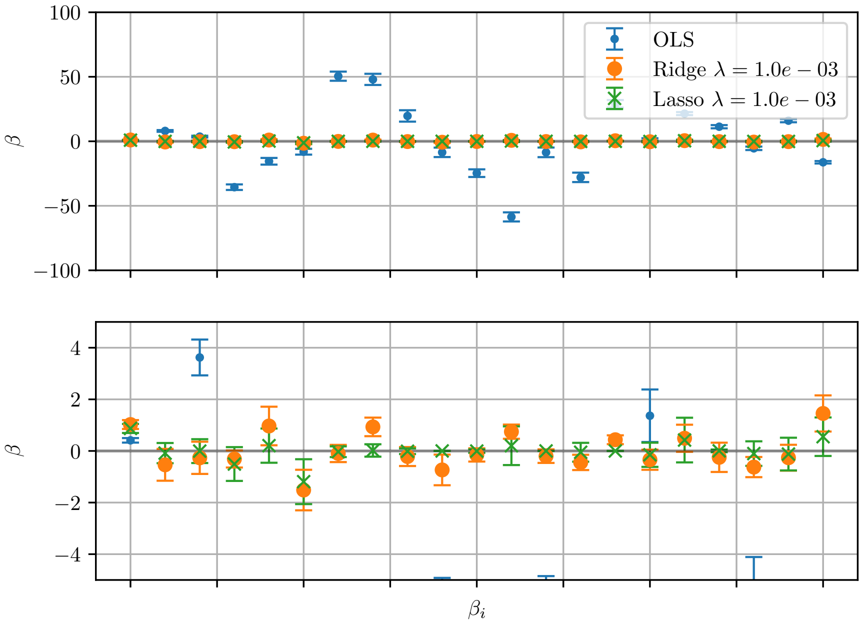


Figure 6: The β parameters of the Franke function for polynomial degree 5, a noise level of ⁄.∀▽∞, ∞ and a $\alpha = 10^{-3}$. The errors are the 95%-CI.

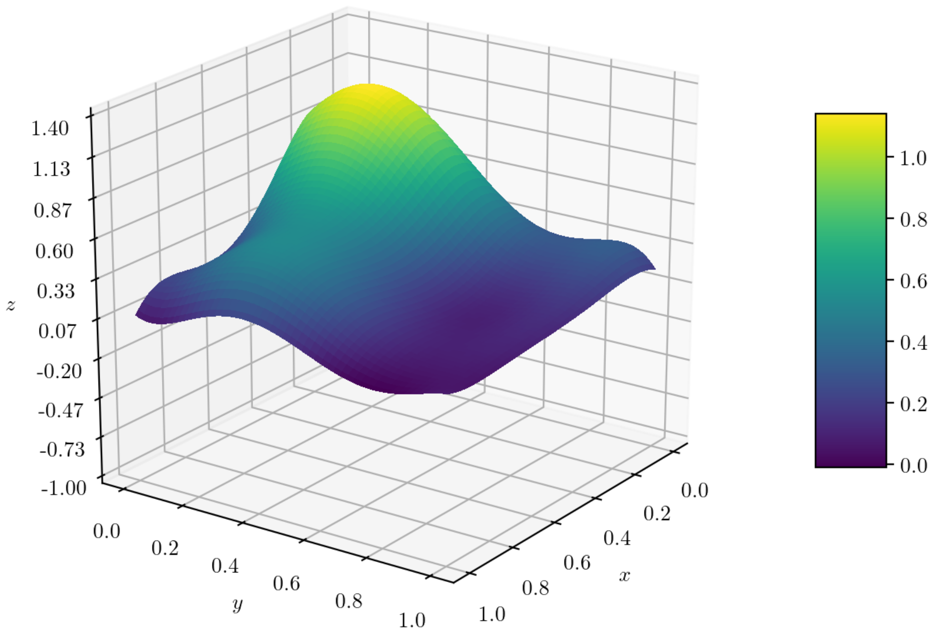


Figure 7: The optimal fit of the Franke function(20), using OLS and polynomial of degree 5.

4.2 Modeling terrain

The terrain we have looked is from a region close to Stavanger in Norway, and can be viewed in figure 8 together with a subset of this region that can be viewed in the sub-figure, 9. We chose a grid of size 100×250. Further, when modeling the terrain, it was normalized by maximal height found in the total dataset.

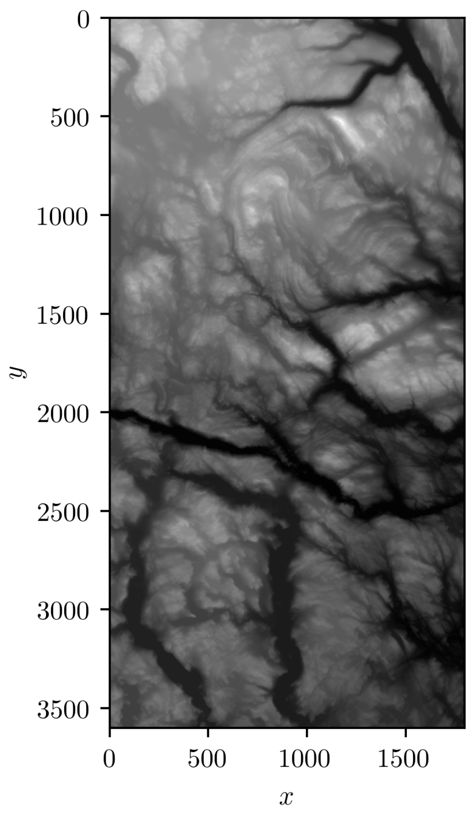


Figure 8: Terrain data from a region close to Stavanger, Norway.

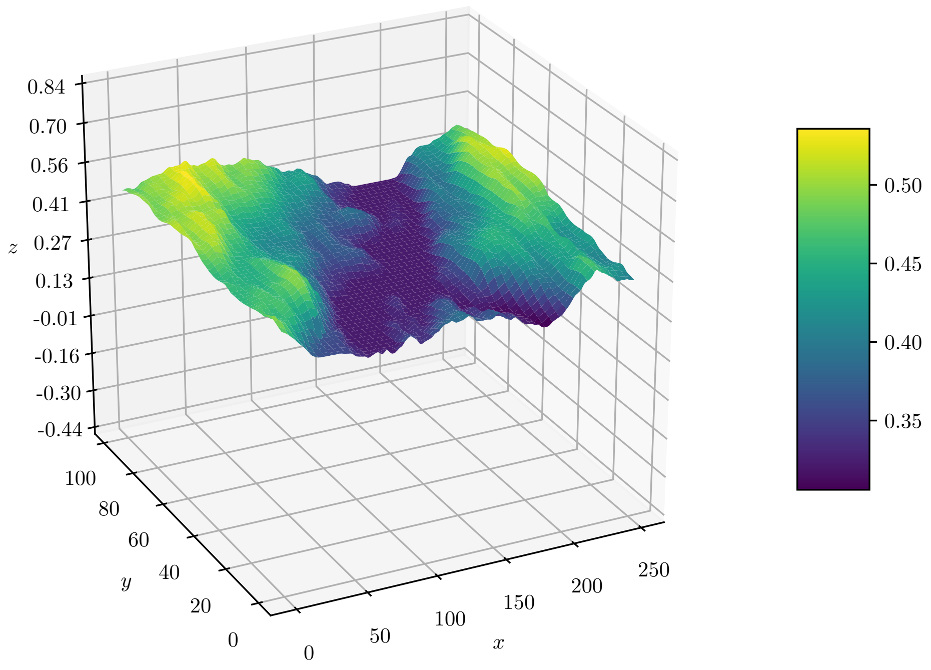


Figure 9: A subset of the region found in figure 8.

The optimal fit method was found by looking at the R^2 and MSE values(16). The result was that the OLS method proved to be the most effective, as summed up in table 2. The figure which these parameters replicates in terms of terrain features can be viewed in figure 11. An example of the optimal fit of the Lasso different Lasso models can be found in table 1.

Score	Degree	R^2	MSE	Bias^2	Var	λ
0.12	8	0.88	$1.72 \cdot 10^{-3}$	$1.72 \cdot 10^{-3}$	$1.61 \cdot 10^{-6}$	10^{-4}

Table 1: Based on the different fitting procedures, these fit parameters were determined to be the best. The polynomial fitted was of degree 8.

Score	Degree	R^2	MSE	Bias^2	Var
$9.0 \cdot -2$	6	0.91	$1.26 \cdot 10^{-3}$	$1.26 \cdot 10^{-3}$	$2.93 \cdot 10^{-6}$

Table 2: Based on the different fitting procedures, these fit parameters were determined to be the best. The polynomial fitted was of degree 6.

$7.95 \cdot 10^{-1}$	$5.37 \cdot 10^{-4}$	$6.15 \cdot 10^{-3}$	$-2.08 \cdot 10^{-4}$
$2.11 \cdot 10^{-6}$	$5.98 \cdot 10^{-5}$	$4.16 \cdot 10^{-6}$	$-4.02 \cdot 10^{-6}$
$7.89 \cdot 10^{-6}$	$-1.30 \cdot 10^{-5}$	$-3.32 \cdot 10^{-8}$	$3.08 \cdot 10^{-8}$
$1.38 \cdot 10^{-9}$	$-1.29 \cdot 10^{-7}$	$3.18 \cdot 10^{-7}$	$1.19 \cdot 10^{-10}$
$-3.57 \cdot 10^{-11}$	$-3.67 \cdot 10^{-10}$	$5.83 \cdot 10^{-10}$	$6.46 \cdot 10^{-10}$
$-2.95 \cdot 10^{-9}$	$-1.58 \cdot 10^{-13}$	$-1.18 \cdot 10^{-13}$	$1.21 \cdot 10^{-12}$
$-1.33 \cdot 10^{-12}$	$-6.44 \cdot 10^{-13}$	$-1.39 \cdot 10^{-12}$	$9.64 \cdot 10^{-12}$

Table 3: The optimal fit β parameters as seen for OLS for a polynomial of degree 6. The β parameters are on row-column order.

The optimal fit of the terrain data using can be seen in figure 11. The best regression method resulted in being OLS regression. The Beta parameters of this fit can seen in figure 3.

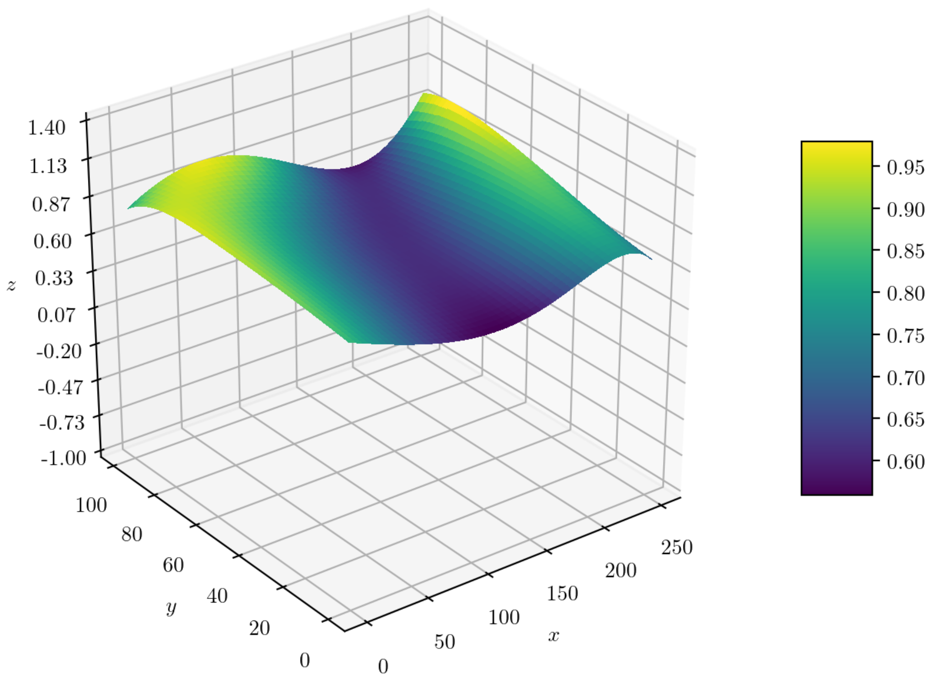


Figure 10: The optimal fit of the terrain data using the Lasso method, for a polynomial of degree 8

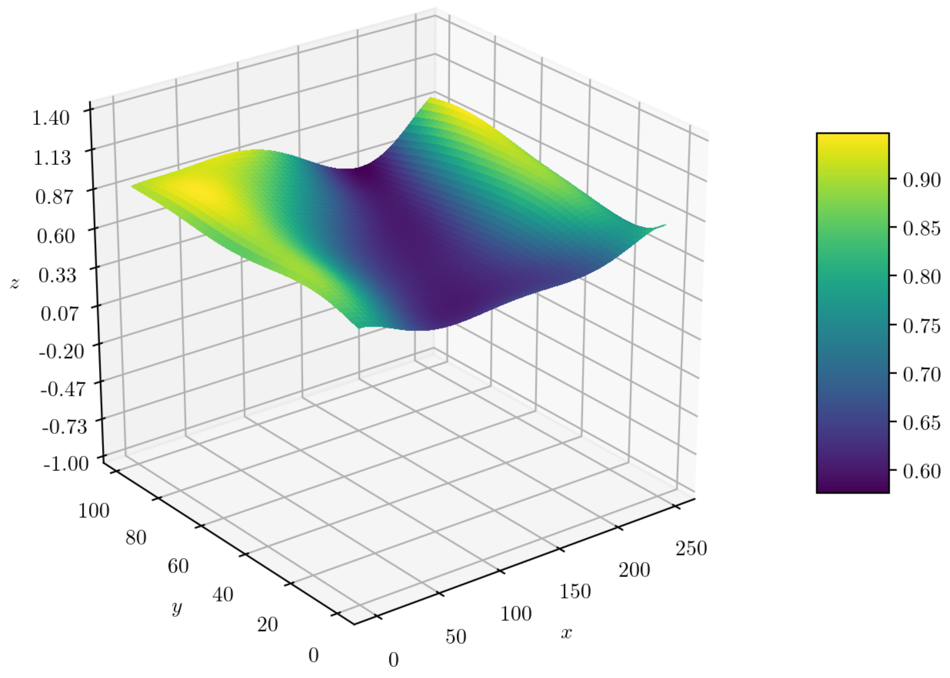
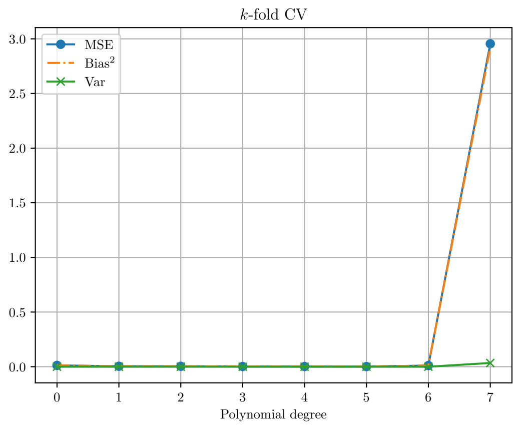
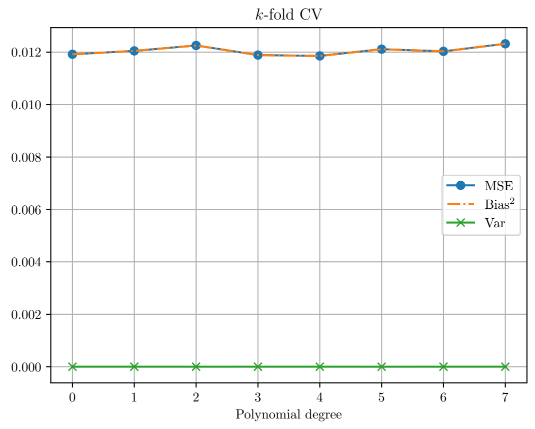


Figure 11: The optimal fit of the terrain data using the OLS method, for a polynomial of degree 6.

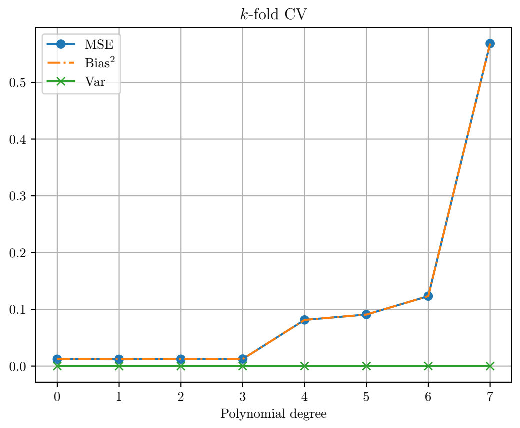
The Bias-Variance of the different terrain fits can be seen in figure 12.



(a) The bias-variance trade off for different polynomial degrees in the fit function OLS on the terrain data.



(b) The bias-variance trade off for different polynomial degrees in the fit function Ridge on the terrain data.



(c) The bias-variance trade off for different polynomial degrees in the fit function Lasso on the terrain data.

Figure 12: Variance-bias plots of the OLS, Ridge and Lasso.

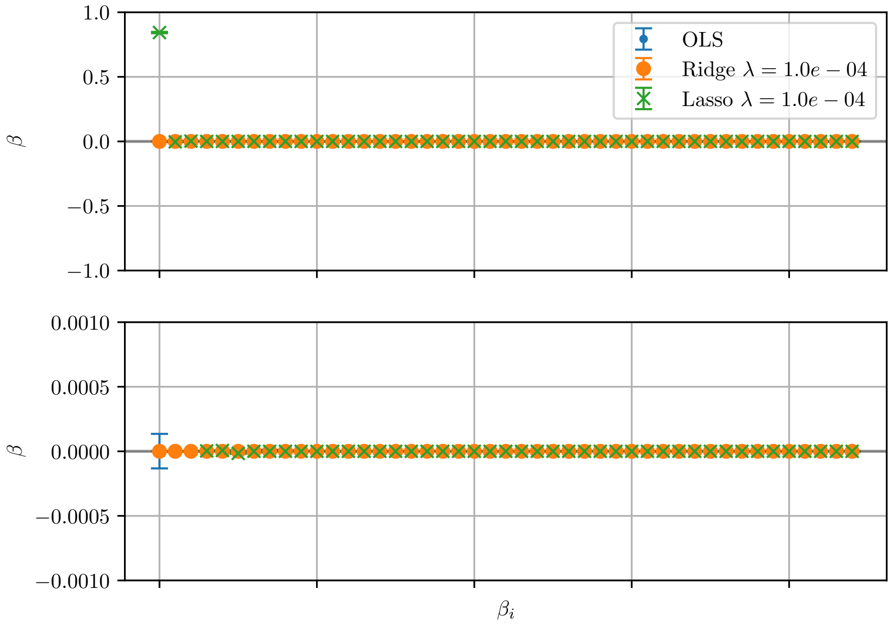


Figure 13: The β values for the optimal fit as seen in 10, plotted with their 95%-CI.

5 Discussion, conclusion and final thoughts

We have now taken a deep dive into the world of linear regression, and it is time to reemerge at the surface and look at what we have learned. We started this journey by using the Franke function as a tool to test the OLS-, Ridge- and Lasso- regression and k-fold CV and bootstrap resampling. What we learned was that the efficacy Ridge and Lasso is greatly dependent on the noise of the data and their λ values, as seen in the figures of 4 and in 3. When looking at figure 3, we also discover that the R^2 value as rather low - never reaches above 0.02. This might indicate that the regression analysis is incorrect for what we have found.

The optimal model which is seen in figure 11 appears to replicate the terrain quite well. In the table 3 we see that the β parameters stretches from around 10^{-1} to 10^{-13}, which of course explains the minuscule β parameters found seen in the figure 6. The β parameters were computed using resampling. As for why the Lasso and Ridge did not outperform OLS for the terrain data, might be explained by the fact that the data patch was not *noise* enough. A better approach to investigate this, would be to randomly sample a set of different terrain patches, and test on each of them to see what gives the best result.

Further, the choice of optimal model through the equation (16) is a rather ad-hoc approach. In the future, an better approach would be to perhaps include the errors in the β parameters in the choice of optimal model.

Personally I would have have needed more time to iron out kinks of the code, and polish what I've written even further. Particularly, I would like to take a closer look at the Lasso regression and figure out why it delivers such low R^2 scores in figure 3. I would also have liked investigate how $MC-CV$ behaves in greater detail as no results for $MC-CV$ were presented here. A more detailed investigation into the resampling techniques would also be interesting. I had more time, a verification of code would also have been implemented more rigorously using unit tests cross checking of data against the methods of SciKit Learn. The dependency of the size of the data set is also something which would have been interesting to probe. Such as, how does the R^2 and MSE behave under different data sets. Further, investigating both the noise levels and the α values with a greater granularity would have been of interest as well.

To summarize, this has been a great exercise into the working of OLS, Ridge and Lasso, and we have seen that the OLS works best when there is little noise, while Lasso appears to for more complicated cases when theres greater variation in noise. Finally, an optimal model for some real terrain data was found. The OLS turned out to be the most accurate one, due to its ability to work around small to medium noise features in a data set. The final fit, as seen in figure13 with $R^2 = 0.91$ and MSE$= 1.26 \cdot 10^{-3}$, which is an acceptable fit considering the scope of this report.

Appendices

.1 The Franke Function

The Franke function as given in [3],

$$\begin{aligned} F(x,y) &= 0.75 \exp\left(-\left(0.25\,(9x-2)^2\right) - 0.25\left((9y-2)^2\right)\right) & (17)\\ &= 0.75 \exp\left(-\frac{1}{49}\left((9x+1)^2\right) - 0.1\,(9y+1)\right) & (18)\\ &= 0.5 \exp\left(-\,(9x-7)^2/4.0 - 0.25\left((9y-3)^2\right)\right) & (19)\\ &= -0.2 \exp\left(-\,(9x-4)^2 - (9y-7)^2\right) & (20)\end{aligned}$$

References

[1] B. Efron and C. Stein. The jackknife estimate of variance. *Ann. Statist.*, 9(3):586–596, 05 1981. doi: 10.1214/aos/1176345462. URL https://doi.org/10.1214/aos/1176345462.

[2] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap.* CRC press, 1994.

[3] Richard Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, Monterey, California: Naval Postgraduate School., 1979.

[4] Morten Hjort-Jensen. Data analysis and machine learning: Linear regression and more advanced regression analysis lecture notes. https://compphysics.github.io/MachineLearning/doc/web/course.html, October 2018.

[5] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[6] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *ArXiv e-prints*, March 2018.

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

[8] Wessel N van Wieringen. Lecture notes on ridge regression. *arXiv preprint arXiv:1509.09169*, 2015.

[9] Mathias Vege. Fysstk4155: Project 1 github repository. https://github.com/hmvege/FYSSTK4155-Project1, October 2018.