Machine Learning

Project 1

Hannah Lovise Ekeberg and Idunn Aamnes Mostue

FYS-STK4155, University of Oslo

October 2019

Abstract

"Two students coming from the outside world of Machine learning, wanting to explore the inside."
— Hannah Lovise Ekeberg, Idunn Aamnes Mostue

This project presents and analyses three linear regression methods - Ordinary least square, Ridge Regression, and Lasso regression. The aim was to find the method of the best fit for data of a known function and some real terrain data. This included finding the suitable parameters of polynomial degree and penalty parameter λ for Ridge and Lasso regression. Through this project we wanted to give a proposal to what model to use, but because of technicalities, we conclude that more testing is necessary before we can give a final proposal.

All code and data used in this project can be found at:

https://github.com/HannahEkeberg/FYS-STK4155/tree/master/Project1

Contents

1. Introduction

In science and engineering it is a common task to want to examine the relationship between some observations and the predictors of the observations. Linear regression models are simple and interpretive methods of describing such relationships. Through linear regression one assume the relationship between the dependent and independent variables to be given by some function and aim to model this function

Through this report we want to analyse three of the most common linear regression methods - Ordinary least square, Ridge regression and Lasso regression - and test them on a known function - The Franke function- as well as some real terrain data. Through statistically estimations of mean squared error (MSE), the R^2 score, and Bias-Variance tradeoff, we try to evaluate the models, and use resampling methods to compare the linear regression models to one another.

This report proceeds in five chapters. Chapter 2 provides theory on the linear regression methods, model assessments and resampling methods used for the analysis. In chapter 3 we introduce the data sets and how we implemented the methods. Further, in chapter 4 we present the main results. And lastly, in chapter 5 we provide a short conclusion of our main findings.

2. Theory

In this chapter we introduce some theoretical background on the most common linear regression models in section 2.1. Further, in section 2.2 we touch upon some of the most popular methods of evaluating the regression models, and lastly in section 2.3 we include some theory on two resampling methods.

2.1 Linear Regression Methods

Regression revolves around the understanding of the relationship between quantities that vary and how one can produce a formula that predicts the value of one variable as a function of other variables. In our case, how one can use a model in finding the best fit for a given set of data. Such models can be linear functions, polynomials, or sinusoids (1), but in this article we focus on the linear regression methods. In general, linear regression gives a model of how a set of data

$$\hat{y}^T = [y_1, y_2, ..., y_{n-1}] \tag{2.1}$$

that we want to explain or model, varies with a given set of variables

$$\hat{x}^T = [x_1, x_2, ..., x_{n-1}]. \tag{2.2}$$

For n cases, indexed $i = 0, 1, 2, ..., n-1$, and *p* explanatory variables called predictors, we get

$$\hat{x}_i^T = [x_i^0, x_i^1, ..., x_i^{p-1}] \tag{2.3}$$

where the predictors tell us something about the complexity of the data we are fitting. Further, we may present the data of $\hat{x}$ on matrix form giving us the *design matrix*, $X \in (n \times p)$

$$X = \begin{pmatrix} 1 & x_0^1 & x_0^2 & \cdots & x_0^{p-1} \\ 1 & x_1^1 & x_1^2 & \cdots & x_1^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1}^1 & x_{n-1}^2 & \cdots & x_{n-1}^{p-1} \end{pmatrix} \tag{2.4}$$

The linear regression model assume a linear relationship between y_i and the predictors p of $\hat{x}$. This gives a set of liner equations on the form

$$y_i = \beta_0 + \beta_i x_{i1} + ... +_p x_{ip} + \epsilon_i \tag{2.5}$$

or on vector form

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.6}$$

where $\boldsymbol{\epsilon} = \boldsymbol{y} - X\boldsymbol{\beta}$ is some term of error concerning the difference between the linearity and the actual data. When we do not have any information about our model, we assume a liner relationship between X and $\hat{y}$ with the linear regression parameter

$$\beta^T = [\beta_0, \beta_1, ..., \beta_{p-1}] \tag{2.7}$$

so that

$$\tilde{\boldsymbol{y}} = X\boldsymbol{\beta} = \begin{pmatrix} 1 & x_{0,1} & x_{0,2} & \cdots & x_{0,p-1} \\ 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1,1} & x_{n-1,2} & \cdots & x_{n-1,p-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix} \tag{2.8}$$

When fitting a model to a given set of data we want to choose a regression parameter β in such a way that the error is minimized, which in turn requires an expression for the error, called a *cost function*. There are many ways of finding the optimal fit for β. Here, we will further investigate the three most common types of linear regression - the Ordinary Least Square, Ridge, and Lasso.

2.1.1 Ordinary Least Square (OLS)

The least square method of regression chooses the regression parameter β so it minimizes the residual sum of squares.

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=0}^{p} x_{ij}\beta_j) \tag{2.9}$$

From this we define its cost function

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{N}(y_i - (X\boldsymbol{\beta})_i)^2 \tag{2.10}$$

And further, from the cost function we can generate the unique solution

$$\hat{\beta} = (X^T X)^{-1} X^T \boldsymbol{y} \tag{2.11}$$

2.1.2 Ridge regression

The main idea with Ridge regression is to introduce a small amount of bias to try to fit our model to the data better, than for OLS. The small amount of bias will cause a drop in variance, and arguably provide a better long term fit for the test data. This method of regression chooses its fitted regression parameter similarly to the method of OLS through minimizing the residual sum of squares, but with an added Ridge penalty. The Ridge estimate is defined

$$\hat{\beta}^{ridge} = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta ij)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here, the ridge parameter $\lambda \geq 0$ controls the shrinkage of the regression parameter. As λ gets bigger it will shrink the regression parameters towards zero. We can then define the cost function

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{N}(y_i - (X\boldsymbol{\beta})_i)^2 + \lambda \sum_{i=0}^{N} \beta_i^2 \tag{2.12}$$

and further generate a unique solution

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y} \tag{2.13}$$

where $\boldsymbol{I}$ is the $(p \times p)$ identity matrix.

2.1.3 Lasso regression

The Lasso regression method is very similar to Ridge, but when looking at the definition of the Lasso estimate (2.14), we see that the Ridge penalty is replaced with some Lasso penalty,

$$\hat{\beta}^{Lasso} = \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=0} p x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{N}|\beta_j| \tag{2.14}$$

The cost function becomes

$$C(\boldsymbol{\beta}) = \sum_{i=0}^{N}(y_i - (X\boldsymbol{\beta}_i))^2 + \sum_{i=0}^{N}|\beta_i| \tag{2.15}$$

For this method, we can not compute a closed form solution for $\hat{\beta}$ as seen for Ridge.

The Ridge parameter can shrink the regression parameter towards zero, but as for the Lasso parameter it can shrink the regression parameters to be equal to zero if λ gets large enough. This means that Lasso also provides a parameter selection, which can lead to a better fit.

2.2 Model assessment

In this section we introduce some of the most common methods of estimating error, namely the Mean Squared Error (MSE), and R^2. Lastly we touch on a method of evaluation called the bias-variance tradeoff.

2.2.1 Mean Square Error (MSE)

The Mean Squared Error provides a measure on how well the model predicts on new data. It is the squared mean of the distance from each data point to the fitted line, defined as

$$MSE(\hat{y}, \tilde{\hat{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{2.16}$$

where

$$\bar{\hat{y}} = \frac{1}{n}\sum_{i=0}^{n-1} y_1$$

is the mean of the estimated values of $\hat{y}$.

2.2.2 R^2

For a regression model, the R^2 score is an estimate of the proportion of the variance for the dependent variables explained by the independent variables. Or in other words for our case - how well our model describe our data. It is defined by

$$R^2(\hat{y}, \tilde{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{\hat{y}})^2} \tag{2.17}$$

again where

$$\bar{\hat{y}} = \frac{1}{n}\sum_{i=0}^{n-1} y_1$$

is the mean of the estimated values of $\hat{y}$.
It range from 0-1, where R^2 =0 indicates that the predicted values equals the mean, where as R^2 =1 indicates that the predicted values equals the observed.

2.2.3 Bias-Variance tradeoff

Two other methods used to evaluate the model is bias and variance. Bias and variance are two sources of error, in which we want to minimize. Bias is a systematic error, and can cause underfitting. Variance is an error which says something about the spread in our data. High variance can cause overfitting as the noise is also fitted. In general, models with a lower bias have a higher variance and opposite, which is called the bias-variance dilemma, as both of those values should be as small as possible. Our data y is expressed as $y = f(x, y) + \epsilon$ which will be shortened to $y = f + \epsilon$ here. Bias is defined as

$$bias[\tilde{y}] = E[\tilde{y}] - E[f] \tag{2.18}$$

$$Var[\tilde{y}] = E[\tilde{y}^2] - E[\tilde{y}]^2 \tag{2.19}$$

The cost function can be expressed in terms of expectation value.

$$C(X, \beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = E[(\mathbf{y} - \tilde{\mathbf{y}})^2] \tag{2.20}$$

We can rewrite equation 2.20, in terms of bias and variance, using a set of different equations

$$E[a+b] = E[a] + E[b] \tag{2.21}$$

$$Var[a] = E[a^2] - E[a]^2 \tag{2.22}$$

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

Adding $E[\tilde{y}] - E[\tilde{y}]$:

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

Arranging the joints:

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

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

From equation 2.21, we can rewrite the above expression as

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

First, $E[\epsilon^2] = \sigma^2$. Secondly, for a deterministic function, $E[f] = f$.

$$(f - E[\tilde{y}])^2 + E[\epsilon^2] + E[(E[\tilde{y}] - \tilde{y})^2] + (f - E[\tilde{y}])E[\epsilon] + 2E[(f - E[\tilde{y}])(E[\tilde{y}] - \tilde{y})] + 2E[\epsilon(E[\tilde{y}] - \tilde{y})] \tag{2.23}$$

Using the expression for bias (equation 2.18) and for variance (equation 2.19), we can rewrite the above expression as

$$(f - E[f])^2 + E[(E[f] - f)^2 + E[\epsilon^2] \tag{2.24}$$

$$(f - E[f])^2 + Var[f] + \sigma^2 \tag{2.25}$$

or for i'th element:

$$E[(y-\tilde{y})^2] = \underbrace{\frac{1}{n}\sum_i (f_i - E[\tilde{y}])^2}_{bias} + \underbrace{\frac{1}{n}\sum_i (tildey - E[\tilde{y}])^2}_{variance} + \underbrace{\sigma^2}_{\text{Irreducable noise}} \tag{2.26}$$

Derivation is more or less copied from (2)

2.3 Resampling methods

It is inconvenient to train and test a model on the same set of data, as overfitting may be a consequence. Therefor, it is more preferable to split the data into a part which trains the model, and a part that tests the model, and predicts the error. To split the data one single time does not necessarily give us the correct error, as it may vary with how the data itself is split citebook. Resampling of data is a way of training and testing the model with different parts of the data as training and test data (i.e. split the data multiple times and estimate the error for each split). This way, we get an prediction error which is not based on one single split of the data, but rather multiple splits and combinations. There are two main methods used in this project to resample the data, namely the bootstrap and K-fold cross validation, of which you can find a short description below.

2.3.1 K-Fold Cross-Validation

The K-Fold Cross-Validation method is based on splitting a dataset into K equal parts, and label the splits $k = 1, 2, ..., K$. Typically, K can be 5 or 10. The model is fit to K-1 parts of the data, and the prediction error of the fitted model is calculated using the last *k*th part not used in train. The resampling is done k times and the prediction error is the mean value for the error for each resampling.

2.3.2 Bootstrap

The Bootstrap method randomly resample a group of data, $f(x, y)$ into n groups (bootstraps). The new groups with resampled data contains the same amount of data, but randomly picked so that the same data $f(x_i, y_i)$ can be listed several times in the same group. Each group is divided into training and test data, and the model is again fitted with the training data and error predicted with the test data. This procedure is repeated for all n groups, and the prediction error is the mean value for the error for each resampling.

3. Data sets and Methods

This section provides a run through of the data and brief description of the methods used in this analysis. Section 3.1 focuses on the Frank function, whereas section 3.2 cover the terrain data.

3.1 The Franke Function

The Frank function is a weighted sum of four exponentials,

$$\begin{aligned} f(x,y) = &\frac{3}{4}exp(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}) \\ &+\frac{3}{4}exp(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}) \\ &+\frac{1}{2}exp(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}) \\ &-\frac{1}{5}exp(-(9x-4)^2 - (9y-7)^2), \end{aligned} \tag{3.1}$$

defined for x,y $\in$ [0,1]. It is a function widely used when testing interpolation - which is what we aim to use it for as well. We use the function with some randomly added noise ϵ with amplitude 0.001, and define the variable $z = f(x, y) + \epsilon$. The design matrix used for the analysis had inputs of x,y and polynomial degree, p. The function for the design matrix can be found on the Github repository under getData.py. It is composed so that x and y are shaped down to one dimensional arrays. The design matrix is made so that the first column consists of 1's. The degree of the matrix is the length of array x or y for rows, with $\frac{1}{2}(p_{i+1})(p_{i+2})$ number of columns.

For the Franke function, the number of possible variables are x, y, xy, x^2 and y^2, giving the value of p=6. And so, the design matrix for the function can be expressed as followed:

$$X = \begin{pmatrix} 1 & x_1 & y_1 & x_1y_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & x_2y_2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n-1} & y_{n-1} & x_{n-1}y_{n-1} & x_{n-1}^2 & y_{n-1}^2 \end{pmatrix} \tag{3.2}$$

The Franke function was tested with various polynomial degrees for OLS, Ridge and Lasso, with and without resampling. Ridge and Lasso was tested with the penalty of $\lambda = 10^{-5}, 10^{-9}$. The Bias-Variance tradeoff was investigated for different polynomial degrees, using bootstrap as well as the mean squared error using K-fold cross validation.

3.2 Terrain data

The terrain data used for this analysis consist of height values over a map of Møsvatn Austfjell, Norway, downloaded from the Github folder of the course FYS4155 at the University of Oslo. Seeing the resolution of the data was so high, we decided to only look at a section of the map and corresponding data, for quicker and more efficient testing and analysing on the computer. We then performed the same analysis to the real terrain data as we did for the Franke function described above.

4. Results and discussion

This section provides a run through of our results from the analysis, presented through tables and graphs. The results of the Frank function follows in section 4.1 and the Terrain data in section 4.2. Discussion of the findings is provided throughout the section.

4.1 Franke Function

Table 4.1 shows overview of the different methods OLS, Ridge and Lasso applied with no resampling. As we can see, generally higher order of polynomial degree gives better MSE and R^2 scores. For higher values of λ, Lasso tends to fit the model better, while as for lower values of λ, Ridge tends to fit the model better. Figures 1 (OLS), 2 (Ridge), 3 (Lasso) shows how the MSE and R^2 varies with higher order polynomial degree for each of the methods. MSE for resampling via K-fold cross validation is represented in table 4.2, and plotted in figures 4, 5 and 6. Here, we clearly see that the mean squared error is generally lower in the case of OLS and ridge. Lasso does not give a representative error, and gives us unexpected results.

The bias variance tradeoff is represented in table 4.2, and supported in figures 7, 8 for OLS and Ridge. As expected, the bias decrease with increasing polynomial degree, and the tendency that with lower values of λ we get a better estimation is still supported. The variance is generally low. For Ridge in figure 9, we see an unexpected behaviour of bias, with very high values, and not decreasing with its characteristic curve. Table 4.3 is an attempt of finding the confidence interval of the optimization parameters β, withtin 95% of the normal distribution. As we can see, the confidence interval tends to increase dramatically with higher polynomial degree. The confidence interval for betas in OLS with no resampling is given in table 4.3, which should give the data for 95% within the normal distribution.

Table 4.1: The table shows mean squared error MSE and R^2 for OLS, Ridge and Lasso on the Franke function with no resampling. Errors are estimated for various polynomial degrees and different values of λ for Ridge and Lasso regression.

Method	**Polynomial degree**	**MSE**	$\mathbf{R^2}$
OLS	1	0.6319	0.0889
	2	0.0269	0.6964
	3	0.0172	0.8053
	4	0.0080	0.9095
	5	0.0041	0.9535
Ridge, $\lambda = 10^{-9}$	1	0.0233	0.6948
	2	0.0158	0.7938
	3	0.0074	0.9028
	4	0.0038	0.9507
	5	0.0020	0.9744
Ridge, $\lambda = 10^{-5}$	1	0.6169	0.0851
	2	1.3074	0.0485
	3	1.3325	0.0543
	4	1.3588	0.0532
	5	1.3789	0.0529
Lasso, $\lambda = 10^{-5}$	2	0.0264	0.6877
	3	0.0171	0.7977
	4	0.0080	0.9051
	5	0.0038	0.9555
	5	0.0020	0.9758
Lasso, $\lambda = 10^{-9}$	1	0.6581	0.0883
	2	1.2957	0.0534
	3	1.1730	0.0661
	4	1.1870	0.0653
	5	1.2279	0.0638

4.2 Terrain data

The original terrain data is represented in figure 10. An overview of the MSE and Bias variance tradeoff is given in figure 11 for OLS and in figure 12 for Ridge. Lasso is not included because our program could not run with Lasso, but supported with the fact that Lasso have an unexpected behavior generally.

Table 4.2: The table shows mean squared error MSE, bias and variance are estimated for OLS, Ridge and Lasso on the Franke function with resampling. Errors are estimated for various polynomial degrees and different values of λ for Ridge and Lasso regression.

Method	Polynomial degree	MSE test	MSE train	bias^2	Variance
OLS	1	0.022389	0.024905	0.022305	0.00008
	5	0.002054	0.002036	0.001947	0.000106
	10	0.000599	0.000274	0.000453	0.000146
	20	0.000741	0.000399	0.000233	0.000508
Ridge, $\lambda = 10^{-5}$	1	0.018681	0.016238	0.018528	0.000153
	5	0.001986	0.002134	0.001920	0.000006
	10	0.000317	0.000335	0.000287	0.000003
	20	0.000213	0.000173	0.000192	0.000002
Ridge, $\lambda = 10^{-9}$	1	0.017070	0.017368	0.016934	0.000135
	5	0.001994	0.002280	0.001876	0.000118
	10	0.000422	0.000336	0.000329	0.000009
	20	0.000243	0.000261	0.000167	0.000007
Lasso, $\lambda = 10^{-5}$	1	0.587896	0.607890	0.587605	0.000290
	5	1.164678	1.164231	1.163021	0.001657
	10	1.084350	1.086010	1.082652	0.001698
	20	1.218479	1.220151	1.216003	0.002476
Lasso, $\lambda = 10^{-9}$	1	0.582270	0.630981	0.581924	0.000346
	5	1.027906	1.026573	1.026457	0.001449
	10	0.906266	0.899241	0.905006	0.001260
	20	1.021432	1.035093	1.018129	0.003303

Table 4.3: The table shows the confidence intervals of estimated mean β-parameters for various polynomial degrees, within 95% of the normal distribution.

Polynomial degree	Average β-value	Confidence interval
1	0.6124	[1.9022 , -0.6774]
2	0.8241	[2.7511 , -1.1029]
3	1.8992	[6.6703 , -2.8719]
4	7.1013	[27.2324 , -13.0298]
5	19.1183	[67.6727 , -29.4361]

Figure 1: The figure shows how R^2 and MSE varies with increasing polynomial degree for OLS with no resampling. The higher the degree, the more MSE approaches zero, and R^2 approaches 1. This is well supported in table 4.1.

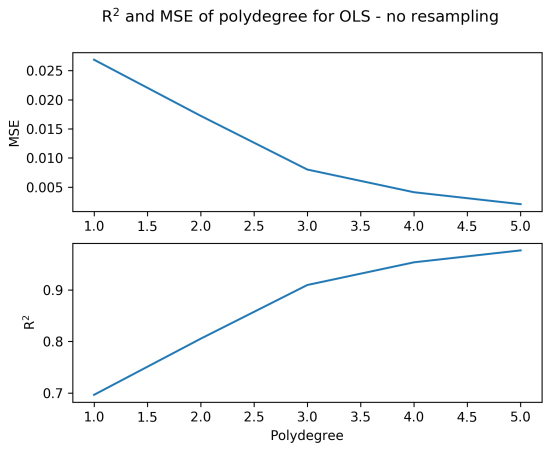


Figure 2: The figure shows how R^2 and MSE varies with increasing polynomial degree for Ridge regression for $\lambda = 10^{-5}, 10^{-9}$ with no resampling. The higher the degree, the more MSE approaches zero, and R^2 approaches 1. We cannot see the effect of different λ clearly in the figure, but table 4.1 supports lower values of λ favours the fit.

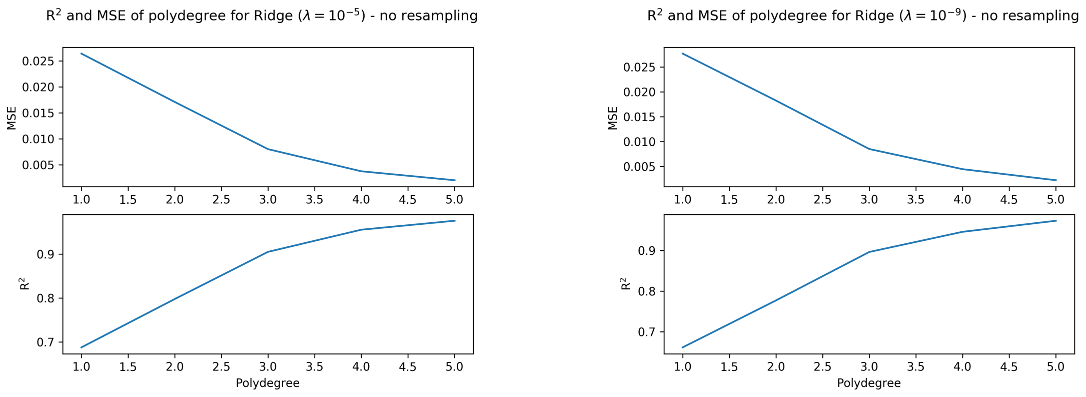


Figure 3: The figure shows how R^2 and MSE varies with increasing polynomial degree for Lasso regression for $\lambda = 10^{-5}, 10^{-9}$ with no resampling. We cannot see the effect of different λ clearly in the figure, but table 4.1 supports higher values of λ favours the fit. The fit is rather unexpected looking, the higher the degree, the more MSE approaches a higher value and R^2 approaches a lower value.

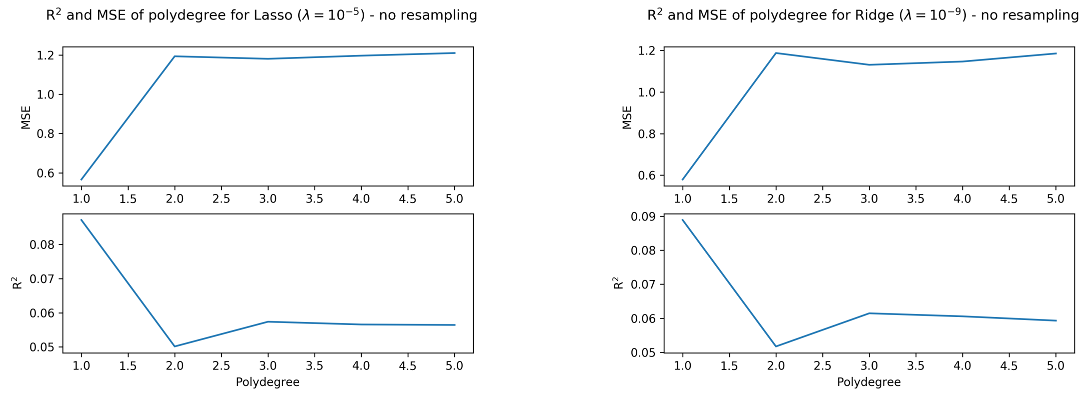


Figure 4: The figure shows how the mean squared error varies with increasing polynomial degree for train and test data in a K-fold cross validation resampling. For polynomial fit up to 15 degrees, the train and test data are more or less identical. For polynomial degrees up to 40, we see tendencies of overfitting in the test data.

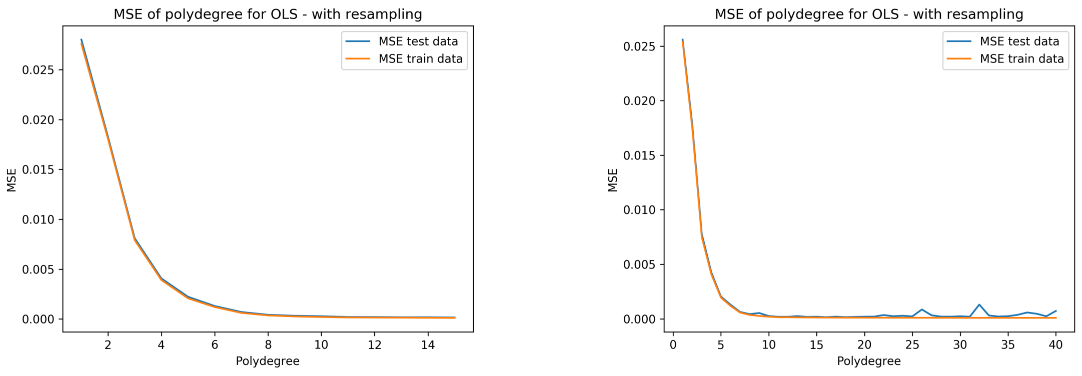


Figure 5: The figure shows how the mean squared error varies with increasing polynomial degree for train and test data in a K-fold cross validation resampling for Ridge ($\lambda = 10^{-9}$). For polynomial fit up to 15 degrees, the train and test data are more or less identical as we saw in OLS, figure 4. For polynomial degrees up to 40, we see tendencies of overfitting around 20 degrees, but from 30 degrees and higher the MSE error of the test data increases more.

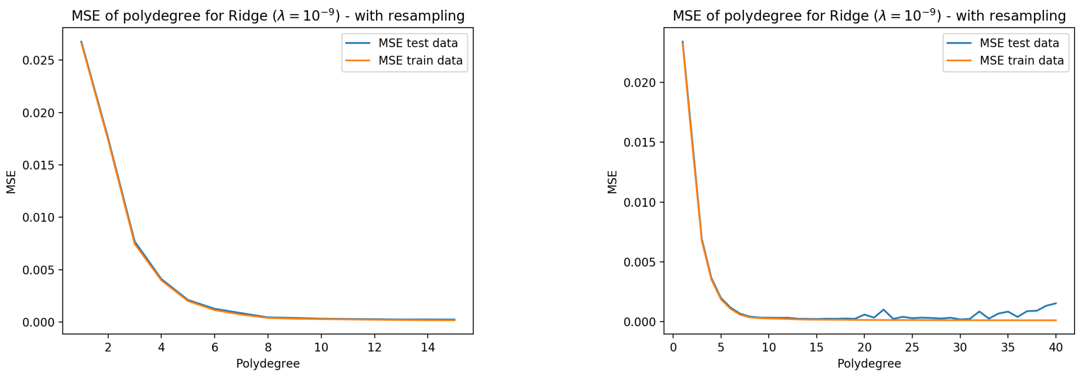


Figure 6: The figure shows how the mean squared error varies with increasing polynomial degree for train and test data in a K-fold cross validation resampling for Lasso ($\lambda = 10^{-9}$). Here, unexpected things are going on, as we saw for the MSE and R^2 with no resample. No further comments will not be necessary.

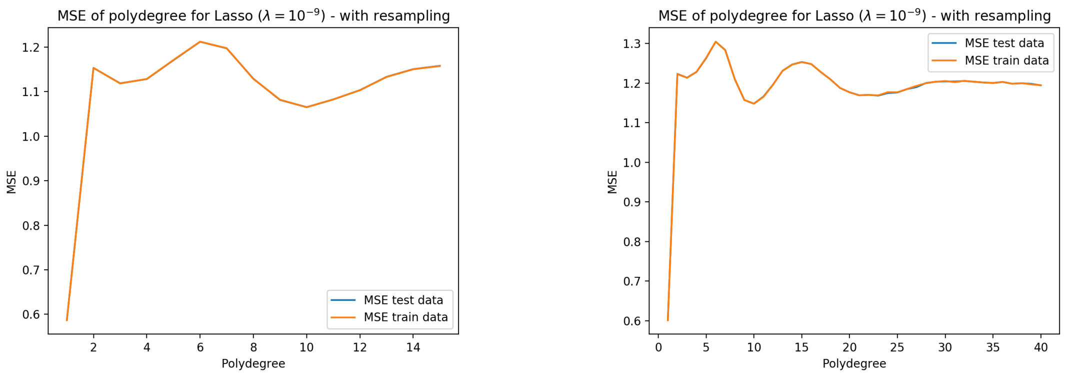


Figure 7: The figure shows the Bias-Variance tradeoff for OLS with polynomial degree up to 20. We get a typical curve for the decreasing bias, but not the typical rise in variance. The variance is low, which is not untypical, which supports that higher order polynomial degree around 11 gives a small bias and small variance.

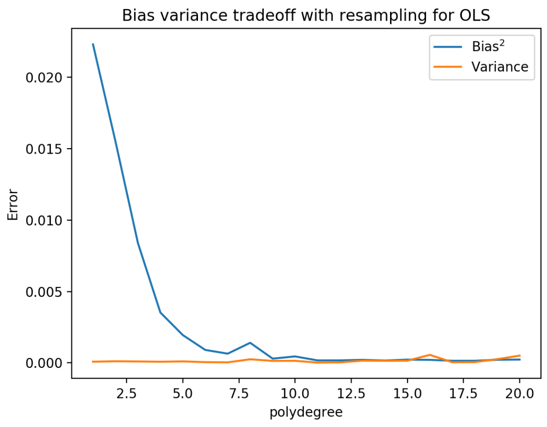


Figure 8: The figure shows the Bias-Variance tradeoff for Ridge with polynomial degree up to 20, with $\lambda = 10^{-5}, 10^{-9}$. We get a typical curve for the decreasing bias, but not the typical rise in variance. The variance is low, which is not untypical, which supports that higher order polynomial degree around 11 gives a small bias and small variance.

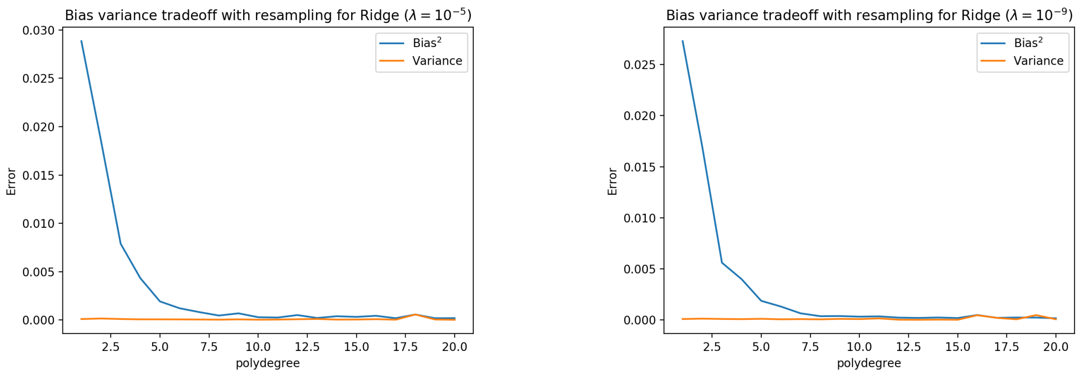


Figure 9: The figure shows the Bias-Variance tradeoff for Lasso with polynomial degree up to 20, with $\lambda = 10^{-5}, 10^{-9}$. The bias is high and varies unexpectedly, while the variance is steadily zero.

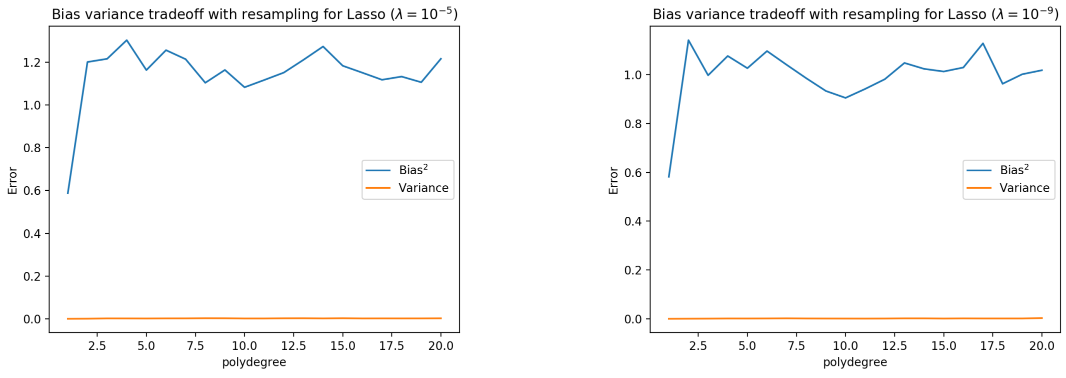


Figure 10: A 250x250 data point image of the terrain Møsvatn, Austfjell, with original terrain data.

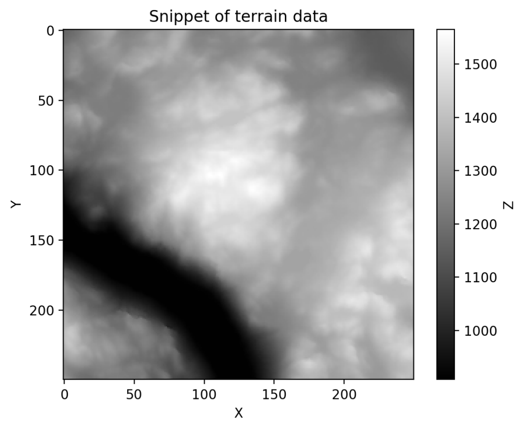
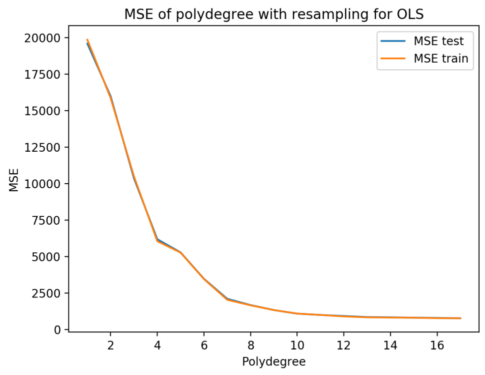


Figure 11: The figure shows mean squared error and bias variance tradeoff for real terrain data for OLS. Errors are genereally high



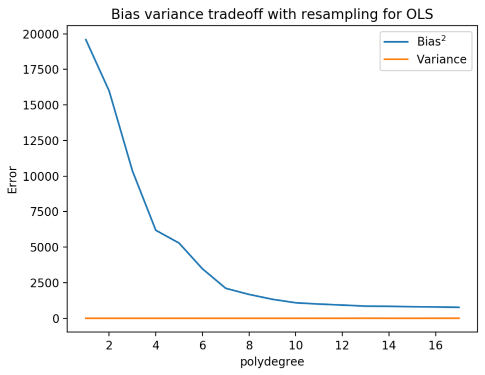
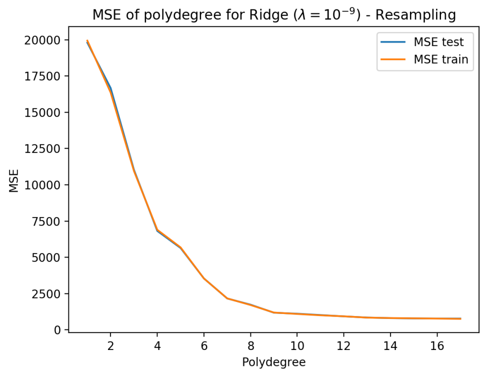
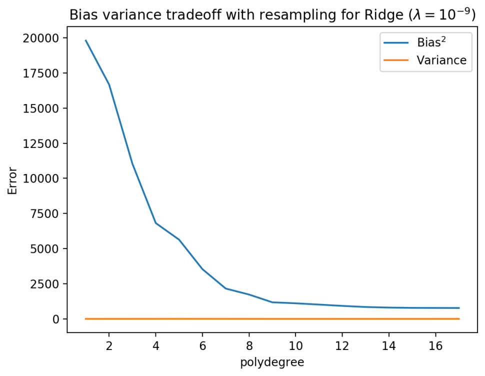


Figure 12: The figure shows mean squared error and bias variance tradeoff for real terrain data for Ridge. Errors are genereally high





5. Conclusion

The aim of this project was to analyze three different linear regression methods and find the best suited estimation values for mean squared error and λ, and test them on a known function and terrain data, then to give a prediction of the best fit for the data. We do not have any clear results which indicates which method gave the best suited estimation values, but looking on table 4.2, Ridge with a lower value of λ (10^{-9} gives the lowest mean squared error. Higher polynomial degree results in better estimation values.

As the program was time consuming to make, we did not have the time to do the tests that we wanted to, modify the program to get the plots and results we expected. Due to this, we do not trust the values and plots generated. In addition, we did not get the program to withdraw the confidence interval of the optimizing parameters β, and was not able to look at the penalty parameter λ, and how different values of λ affected the error analysis. Therefore, we conclude that in order to give a prediction on what model fits the data best, we need to do more tests and modify the program to run more efficiently to be able to produce more interpretive results.

References

C. Lay, R. Lay, McDonald, (2016). *"Linear Algebra and its Applications", 5th edition*. University of Maryland, Pearson. Chapters 6.6.

Wikipedia. *`Wikipedia,https://en.m.wikipedia.org/wiki/Bias\T1\textendashvariance_tradeoff` 4.10.2019.*

Hastie, Tibshirani, Friedman, (2017). *"The Elements of Statistical Learning. Data mining, Interference and Prediction", Second edition.* Standford University, Springer. Chapters 2, 3 and 7.