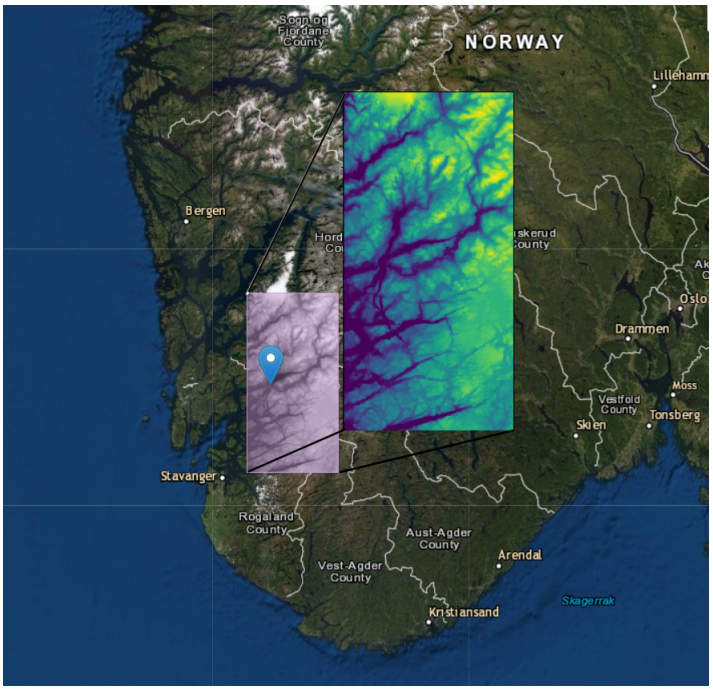
Regression analysis and resampling methods

Jens Due, Mohamed Ismail & Oliver Hebnes

October 7, 2019



Abstract

In this project, three regression methods are developed and tested with the aim of approximating two different data sets and generating a model that fits these data. The first data set is generated from the Franke function, and the second is terrain data from an area in the south-western part of Norway. The regression methods used are the ordinary least squares, the Ridge-regression and the LASSO-regression. For analyzing the quality of the models, the mean squared error and R2-scores are calculated and plotted as a function of increasing model complexity, as well as the bias, variance and confidence intervals. It is shown that the models trained on the data from the Franke function has close to ideal values for the mentioned quality-parameters, while the models do not perform as well on the terrain data. This is concluded to be because of the high complexity of the terrain data, and inability to create a model with corresponding complexity.

Contents

I Introduction

The ability to learn from our mistakes is of huge importance to human beings. This is how the world has been formed, by trial and error. Humans do not possess the ability to program each other, but since newer, modern times, we have discovered tools that could have been seen as magic for only a century ago. The pure combination of programming and statistics form this magnificent basis as we dive into three different regression types while trying to avoid dangers as getting a too much biased model or using too few data points. The purpose of the regression analysis is to try and model the infamous Franke-function and, later on, a real topographic area in the north named Norway. The possibilities of this application are limitless and it inspires a whole new generation of data scientists to achieve what we did not dare to dream about in the past.

II Theoretical Methods and Technicalities

i Ordinary Least Square

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{1}$$

This linear regression model assumes that the regression function ($E(Y|X)$ is a reasonable approximation to the linear model. Here the β_j are unknown parameters and the variables X_j can come from different sources.

The residual sum of squared errors (RSS) is deviations predicted from actual empirical values of data. It is a measure of the discrepancy between the data and an estimation model. A small RSS indicates a tight fit of the model to the data.

It is used as an optimality criterion in parameter selection and model selection.

$$RSS = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{2}$$

$$= \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} X_j \beta_j)^2 \tag{3}$$

One can minimize RSS to obtain the unique solution

$$\hat{\beta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{4}$$

with the design matrix

$$\mathbf{X} = \begin{bmatrix} x_{00} & x_{01} & x_{02} & \dots & x_{1m} \\ x_{10} & x_{11} & x_{12} & \dots & x_{2m} \\ & \dots\dots\dots\dots & & & \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nm} \end{bmatrix} \tag{5}$$

giving the fitted values from the training inputs

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\beta} = \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{6}$$

This is indeed the solution we are looking for in ordinary least squares.

ii Ridge regression

Ridge regression shrinks the regression coefficients β by imposing a penalty on their size.

$$\hat{\beta}^{ridge} = \text{argmin}_\beta \Big\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Big\} \tag{7}$$

Here λ is a complexity parameter that controls the amount of shrinkage. The larger the value, the greater the shrinkage. The ridge solution are not equivariant (acts symmetrically on another symmetric space), so one needs to standarize the inputs before solving this equation.

This can also be written in the RSE-form, but with a size constraint on the parameters.

$$\hat{\beta}^{ridge} = \text{argmin}_\beta \sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j \Big)^2 \tag{8}$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t \tag{9}$$

There is a one to one correspondance between λ and t.

When there are too many correlated variables in a linear regression model, their coefficients can become poorly determined and exhibit high variance. A wildly large positive coefficient on one variable can be canceled by a similarly large negative coefficient on its correlated cousin. But by imposing a size constraint on the coefficients, as in (9), the problem decreases in importance.

The ridge solution can also be written in matrix form.

$$RSS(\lambda) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta) + \lambda \beta^T \beta \tag{10}$$

with the ridge regression solution as

$$\hat{\beta}^{\text{ridge}} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{11}$$

iii Lasso regression

The LASSO (Least Absolute Shrinkage and Selection Operator) regression is also a shrinkage method like Ridge, but there is a small difference. Ridge regression wants to minimize the loss function, and to do so we add a hyperparameter λ to minimize the parameters that does not matter that much. LASSO has, however, the ability to make the coefficients zero and totally exclude them if they are not relevant for the model.

The implementation of LASSO was done by using a module from the scikit-learn package.

In addition, LASSO also depends on another parameter called the learning rate γ, and this parameter will be optimized by the sickit learn package using various methods, but as this is not an important subject of this project, we will not elaborate further.

iv Confidence interval

To calculate the confidence interval of the parameter $\boldsymbol{\beta}$ one would have to first calculate the variance of $\boldsymbol{\beta}$:

$$Var(\boldsymbol{\beta}) = (\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}})^{-1}\sigma_{\boldsymbol{\epsilon}}^2$$

The standard deviation is defined as the square root of the variance, so the standard deviation of $\boldsymbol{\beta}$ would then be:

$$\sigma_{\boldsymbol{\beta}} = \sqrt{(\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}})^{-1}\sigma_{\boldsymbol{\epsilon}}^2} = \sqrt{(\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}})^{-1}}\sigma_{\boldsymbol{\epsilon}}$$

This is a special case where we know the variance of $\boldsymbol{\beta}$, but for the general case, this may not be known. One approach is then to approximate the variance with the mean squared error for $\boldsymbol{\beta}$.

$$\begin{aligned} Var(\boldsymbol{\beta}) &\approx MSE_{\boldsymbol{\beta}} \\ \sigma_{\boldsymbol{\beta}} &\approx \sqrt{MSE_{\boldsymbol{\beta}}} \end{aligned}$$

Then to find a 95% confidence interval for $\boldsymbol{\beta}$, one needs to use the formula:

$$\boldsymbol{\beta} \pm z \times \sigma_{\boldsymbol{\beta}}$$

Where z is the 95th percentile of the normal distribution. This corresponds to $z = 1.96$.

The theory can be found in the textbook of Hastie et al. [2].

v Bias-Variance Tradeoff

We have assumed that our data can be represented by an unknown function with the addition of some noise $\boldsymbol{\epsilon}$.

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}.$$

Where $\boldsymbol{\epsilon}$ is normally distributed with a mean equal to zero and standard deviation equal to σ^2.

Furthermore, we also assume that the function $f(\boldsymbol{x})$ can be approximated to a model $\tilde{\boldsymbol{y}}$, where the model is defined by a design matrix $\boldsymbol{X}$ and parameters $\boldsymbol{\beta}$.

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$$

The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the mean squared error (MSE) via the so-called cost function

$$C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = E\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right].$$

One can show that the cost function can be rewritten as

$$E\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] = \frac{1}{n}\sum_i (f_i - E\left[\tilde{\boldsymbol{y}}\right])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - E\left[\tilde{\boldsymbol{y}}\right])^2 + \sigma^2.$$

By simply using these relations

$$\begin{aligned} E[\boldsymbol{y}] &= \boldsymbol{f} \\ E[\boldsymbol{\epsilon}] &= 0 \\ Var(\boldsymbol{y}) &= Var(\boldsymbol{\epsilon}) = \sigma_\epsilon^2 \end{aligned}$$

Since the variance of $\boldsymbol{y}$ and $\boldsymbol{\epsilon}$ are both equal to σ^2. The mean value of $\boldsymbol{\epsilon}$ is by definition equal to zero. In addition, the function $\boldsymbol{f}$ is not a stochastic variable, and the same argument can also be used for $\tilde{\boldsymbol{y}}$. Then we plug in the definition for $\boldsymbol{y}$ into the cost function:

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

Adding and subtracting $E[\boldsymbol{y}]$ we get:

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

And simply by using the relations mentioned above concerning the expectation value for $\boldsymbol{y}$ and the variances for both $\boldsymbol{y}$ and ϵ, the cost function can be rewritten to:

$$\begin{aligned} E[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])^2] + Var(\tilde{\boldsymbol{y}}) + \sigma_\epsilon^2 \\ E\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \frac{1}{n}\sum_i (f_i - E\left[\tilde{\boldsymbol{y}}\right])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - E\left[\tilde{\boldsymbol{y}}\right])^2 + \sigma_\epsilon^2. \end{aligned}$$

The full derivation can be found in the appendix.

The first term on the right hand side is the squared bias, the amount by which the average of our estimate differs from the true mean. The second term represents the variance of the chosen model and finally the last term is the variance of the error $\boldsymbol{\epsilon}$.

The diagram above illustrates the effect the complexity of a model has on the MSE, bias and variance. The more complex model one has, the lower the bias becomes, and the higher the variance becomes. An ideal model would be one that simultaneously achieves low variance and low bias.

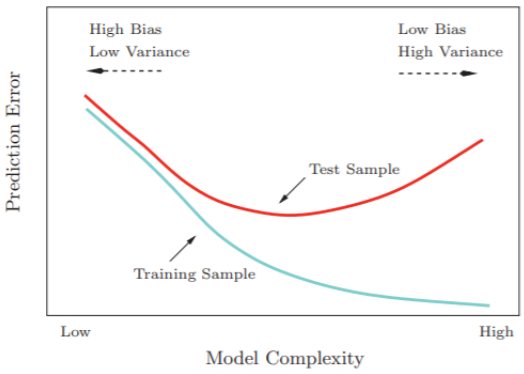


Figure 1: Test and training error as a function of model complexity [2]

vi MSE and R2-score

Two of the main methods for determining how well our model fits the data, will be evaluating the MSE (mean squared error) and the R2-score. They are defined as

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

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

For a perfect fit, these would be valued at 0 for the MSE, and 1 for the R2-score. These are the values our model should approach in order to be a good approximation of the data, and our goal is to make sure this is the ca se.

vii Resampling

Resampling methods are an indispensable tool in modern statistics. They involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model [1].

i K-Fold Cross Validation

K-fold cross validation is a resampling technique where a given data set is divided into a K number of subsets/folds. One after another, each fold will play the role of the test set while the rest are used to train the model, as visualised in the diagram below where $K = 5$. This procedure ensures that the entire data set is being used both as training and testing data in a balanced manner.

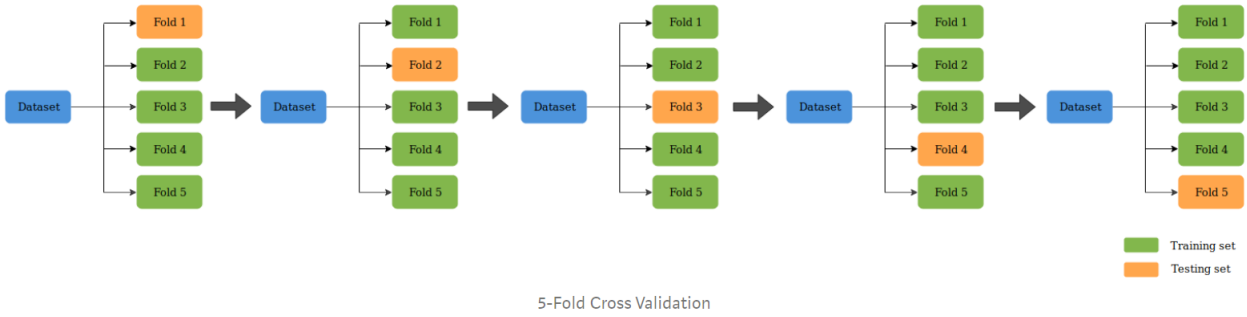


Figure 2: Diagram visualising the 5-fold Cross Validation [3]

III Data Sets

i Franke's Function

Our first data set is generated from Franke's function. It is defined as

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

This is a function which has been widely used when testing various interpolation and fitting algorithms.

In this project added stochastic noise was to Franke's function before fitting to it. The included noise was normally distributed with mean equal to zero and standard deviation set to one, $N(0,1)$.

A plot of the Franke function without the noise can be seen in the figure 3.

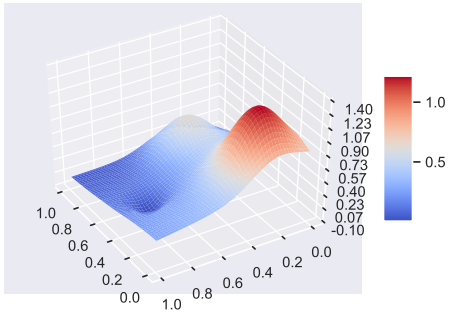
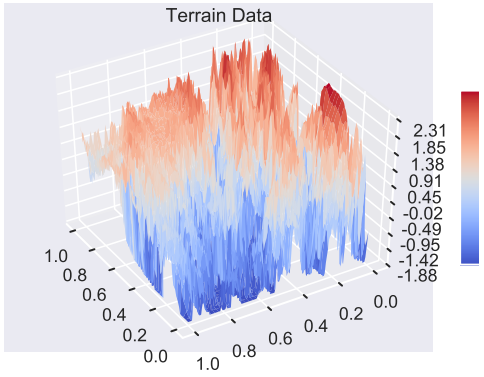


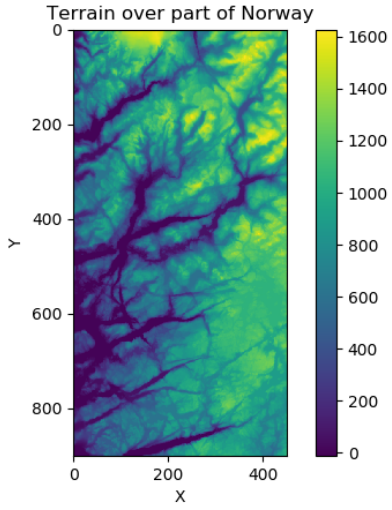
Figure 3: The Franke function plotted for $0 \leq x, y \leq 1$

ii Terrain Data

Our second data set is terrain data from an area on the south-eastern part of Norway, not far from the city of Stavanger. The data is sourced from the U.S. Geological Survey [4]. A visual representation of the data is shown in the figure 4.



(a)



(b)

Figure 4: The terrain data visualised in both two and three dimensions. Figure (a) has been normalised while the z-value in (b) has real value. The data is gathered just outside of a place in the south-west part of Norway called Stavanger. Do you notice the steep fjords and the calm waters?

IV Implementation

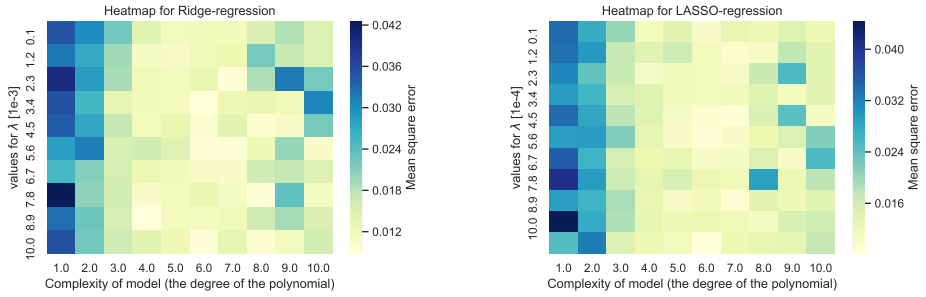
Programs used in this project can be found in our GitHub repository [5].
These programs are inspired by the codes found in the course's GitHub repository [1].

V Results

i Data from Franke's function

i Choosing hyperparameter λ

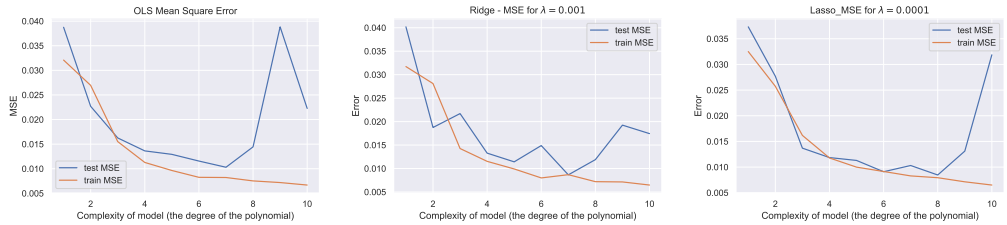
From figure 5 it is clear that the choice of λ is not as crucial as choosing the correct polynomial degree to minimise the mean square error. We can see that several λ-values give satisfactorily small MSE, thus we choose $\lambda = 10^{-3}$ & $\lambda = 10^{-4}$ for further analysis with Ridge- and Lasso-regression.



(a) (b)

Figure 5: Heatmaps showing different hyperparameters λ as a function of model complexity and mean squared error for (a) Ridge-regression, and (b) LASSO

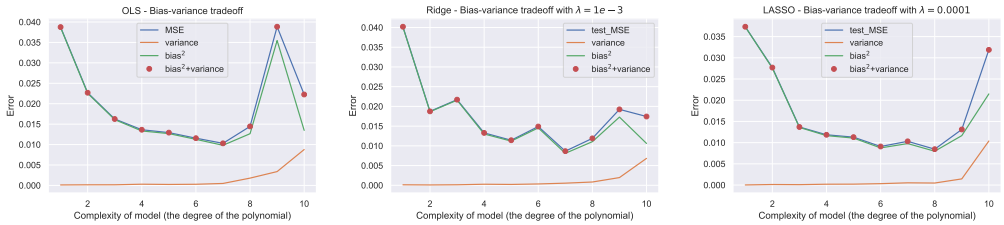
ii Train and test model



(a) (b) (c)

Figure 6: The mean squared error of the testing and training data as a function of model complexity on Franke's Function for (a) OLS, (b) Ridge and (c) LASSO-regression.

iii Bias and variance



(a) (b) (c)

Figure 7: The mean squared error, bias and variance as a function of model complexity on Franke's Function for (a) OLS, (b) Ridge and (c) LASSO-regression.

iv Confidence Interval

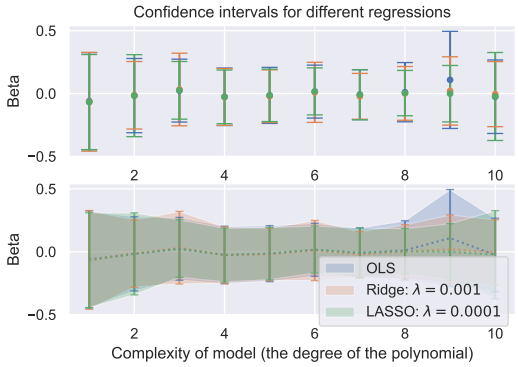
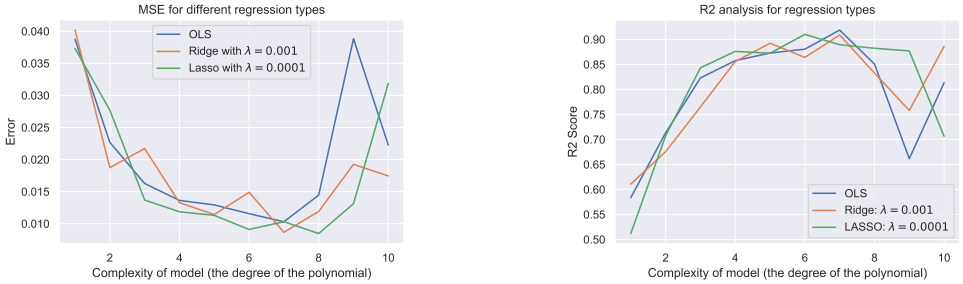


Figure 8: Confidence intervals for the regression coefficients β for all three regression methods. For polynomial degrees 4-7, the intervals are quite small and overlapping. Outside this range, the intervals are shown to be larger.

v Regression analysis



(a) (b)

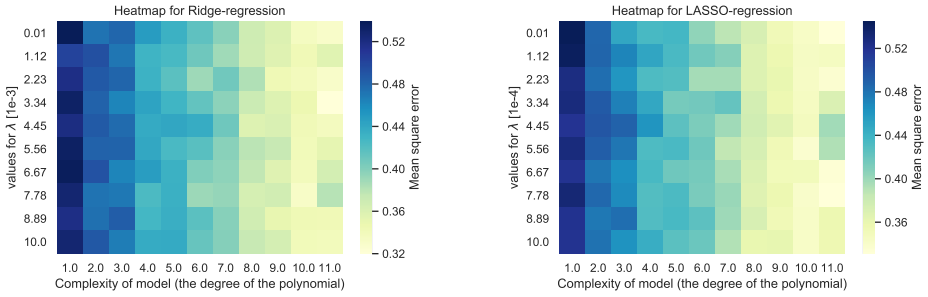
Figure 9: The mean squared error and the R^2-score as a function of the model complexity on Franke's function with OLS, Ridge & LASSO.

ii Terrain data

Here, we repeat the above analysis on the new data set from the terrain data.

i Choosing hyperparameter λ

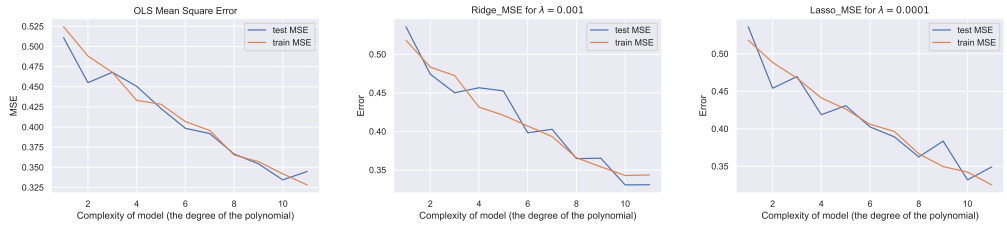
As mentioned before, choosing a hyperparameter λ is normally of great importance for Ridge and LASSO. However, figure 10 shows similar trends as figure 5, that the choice of λ is not very important, and thus we will choose the same values as was done with Franke's function for further analysis on the terrain data.



(a) (b)

Figure 10: Heatmaps showing different hyperparameters λ as a function of model complexity and mean squared error for (a) Ridge-regression, and (b) LASSO
for Franke's function.

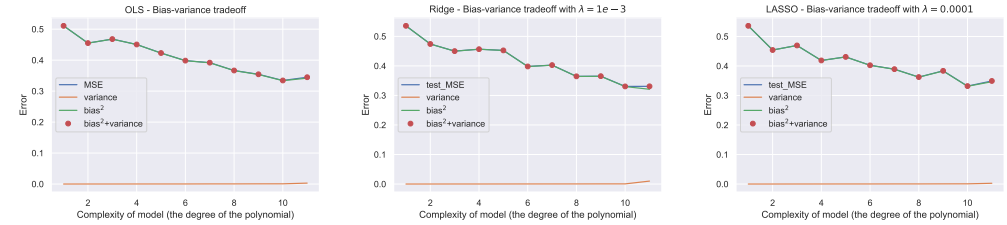
ii Train and test model



(a) (b) (c)

Figure 11: The mean squared error of the test and train data as a function of model complexity on the terrain data for (a) OLS, (b) Ridge and (c) LASSO-regression.

iii Bias and variance



(a) (b) (c)

Figure 12: The mean squared error, bias and variance as a function of the model complexity on the terrain data for (a) OLS, (b) Ridge and (c) LASSO-regression.

iv Confidence Interval

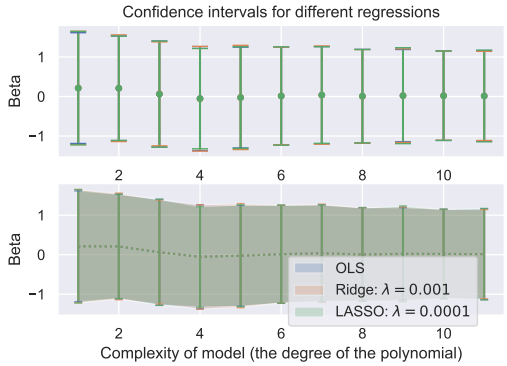
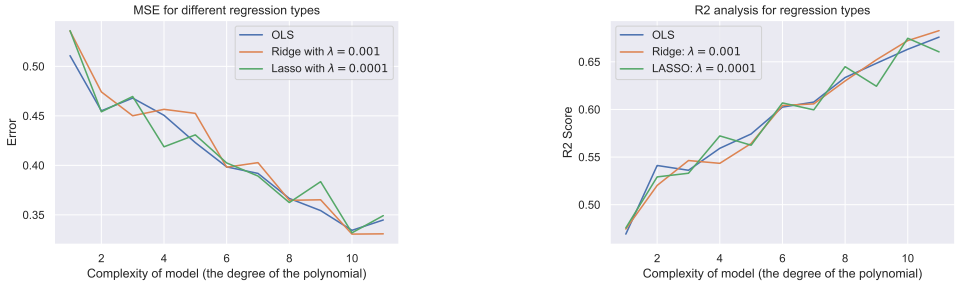


Figure 13: Confidence intervals for the regression coefficients β for the terrain data. We see the intervals essentially completely overlap for all three regression methods.

v Regression analysis



(a) (b)

Figure 14: The mean squared error and R^2-score as a function of the model complexity on the terrain data with OLS, Ridge & LASSO.

VI Discussion

i Franke's Data

As discussed earlier in the text, both Ridge and LASSO depend on the hyperparameter λ for minimising the mean squared error. And this is clearly illustrated for both Ridge and LASSO in figure 7, where the MSE has been plotted for various λ as a function of the complexity of the model.
The figure also tells us that there are no specific λ that minimises the error, and that a lot of different values of λ minimise the error equally.

From figure 6 we clearly observe that the training data is performing better than the testing data for all the regression methods. As the complexity increases, the mean square error decreases, and will in theory decrease until there will be no error for the training data. However, this is not the case for the testing data. As we train our model on the training data, we will make an increasingly tight fit to it which may cause overfitting. We do want to decrease our training mean square error, but we would also like to use our model on independent test sets and still have the error as low as possible. We will not be able to do this if we overfit the model to the training data, and this is shown in the figure where the training data gets a tighter fit, but the testing data does not experience the same for increasing polynomial degrees.

By reading off the results of figure 7, we see that the bias is the main contribution to the MSE up until a certain complexity of the models for all the regression methods. At a certain point, however, the variance drastically increases. These results reflect the behaviour we discussed earlier in the report, where the complexity of the model affects the bias and variance.

The complexity of the model that minimises the bias and the variance is not the same for all the regression models. OLS achieves this at polynomial degree 7, as well as Ridge with $\lambda = 10^{-3}$, while LASSO with $\lambda = 10^{-4}$ does it at polynomial degree 8.

With an increasing number of data points for Franke's function, it was found that the model became more stable. Looking back at the figure 5 one can easily see deviations of the model. One example is the ordinary least square method in figure 7a where the mean square error peaks at polynomial degree 9. One can look at the corresponding confidence interval for beta in figure 8, where the interval peaks as well at polynomial degree 9. This was not the case when the number of data points were increased, but on the other hand we did not get an increase in either variance, or a substantial decrease in bias for a polynomial degree up to 10. When increasing the polynomial degree up to 15, the values were exploding, both with a small and large amount of data points. This can easily be visualised in the jupyter notebook in our GitHub repository [5]. One of the reasons behind this might be lack of numerical precision. Another reason can be instability in the implemented algorithms.

From figure 9a the lowest mean square error is achieved at polynomial degree 7 for Ridge and polynomial degree 8 for LASSO.
At each time the program is executed, the data that becomes the test data changes, and the mean squared error and R^2-score changes along with each execution.
This also indicates that the best regression type will change every time the program is run. Ideally

the mean square error should be 0 and the R^2-score should be 1 for a perfect fit, but as seen from the results, this is not the case.

On the other hand it is the ordinary least square method that results in the biggest R^2-score for the polynomial degree of 7. Nonetheless, the difference of each R^2-score at polynomial degree 7 is not crucial in the analysis, and will experience modifications for every run of the program.

ii Terrain Data

Figure 10 is similar to 5, but with some differences. The complexity of the terrain data will demand a higher polynomial degree for low mean square errors, and once again, the hyperparameter λ is not as important as the complexity. The mean square error is a function of both bias and variance, but for the lambdas and degrees of polynomial that were simulated, we did not see a substantial increase in mean square errors. For higher polynomial degrees, it was expected that the bias-variance trade-off that was seen for lower polynomial degrees for Franke's function would present itself, but unfortunately, due to both numerical instability and precision, the numbers were exploding.

There was not observed any clear signs of overfitting when observing the figure 11, due to an improvement of the mean square error for both training and testing data. This is further strengthened by looking at figure 12 where the only contribution for the mean square error is the bias. However, this might be a sign of underfitting, which might indicate either that the model has not trained enough, or it is not using a high enough polynomial degree.

It is of interest to train on a large scale of data, and test on the remaining small parts of the data. If there is not enough data points to train on, one will not get a general enough model that will work on data different from the training data. On the other hand, if there is too much data with high complexity, as is the case with the terrain data, the difficulty increases for developing and enhancing a model that will work on a given data set. This demands stronger optimization of code and more powerful machinery.

In a highly varying landscape such as our terrain data, the calculated confidence interval for the regression coefficients β will behave accordingly. This behaviour is seen in figure 13 for all regression types where significant intervals indicates a not very good fit. This is expected because of the varying data.

The R2 scores and the mean square errors in figure 14 shows that there is not necessarily one regression type that exceeds, but that every one has their pros and cons when they work in their best environment. It was expected that the regression type with the lowest mean square error would experience the highest R2 score, but our results for both Franke's function and the terrain data contradict this hypothesis. Nonetheless, we are seeing a much better fit of the testing data when comparing figure 9 and figure 14. As for Franke's function, the mean square error is 1/30 of the mean square error of the terrain data, while the R2 scores for Franke's function reach values above 0.90 against the terrain data's 0.67. While the mean square error is not necessarily an indication of a model that will do well on numerous different data sets, because a model can be a fantastic model with bad values of mean square error, a R2 score gives a score based

independently on the amount of data sets. Thus, a higher R2 score will be a better model.

VII Conclusion

In this project, we have implemented the regression methods OLS, Ridge and LASSO for data from Franke's function and topographic data from southern Norway. The results show no clear indication of which model fitted the data best, regarding the Franke function, with respect to the mean squared error, where LASSO had the lowest MSE, and the R^2-score, where OLS had the highest score. This discrepancy may have been caused of possible errors in our code which could have given us these contradicting results. While for the terrain data, the Ridge-regression model is slightly better than OLS and LASSO, in respect of the mean squared error and the R^2-score. But as mentioned before these results may vary each time the program is executed, because of the random choice of test data. The art of balancing the eternal bias-variance trade-off while struggling with a huge amount of calculations, while only at a apprentice level, has yielded us great experience and a unique insight into data handling, regressions and resampling.

As an evaluation of our work through the project we found some potential in writing down diaries during the weeks, since there were many times we spent a tiny amount of a lifetime on figuring out what we had done earlier. In addition, using time- and CPU-efficient packages such as scikitlearn will provide more stable and robust machinery, which will yield good result for higher complexity of polynomials.

References

[1] Morten Hjorth-Jensen, Github Repository, https://github.com/CompPhysics/MachineLearning/tree/master/doc

[2] Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, The Elements of Statistical Learning, Springer https://www.springer.com/gp/book/9780387848570.

[3] Medium, Data Driven Investor, K-Fold Cross Validation https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833

[4] U.S. Geological Survey, EarthExplorer https://earthexplorer.usgs.gov/

[5] GitHub repository containing all code and plots, as well as a Jupyter notebook for easy reproducibiliy https://github.com/ohebbi/FYS-STK4155/tree/master/project1

VIII Appendix

i Derivation of The Bias -Variance Tradeoff

$$
\begin{aligned}
E[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= E[(\boldsymbol{f}+\boldsymbol{\epsilon}-\tilde{\boldsymbol{y}})^2] \\
E[\boldsymbol{y}] &= f \\
E[\epsilon] &= 0 \\
Var(\epsilon) &= \sigma_\epsilon^2
\end{aligned}
$$

Adding and subtracting $E[\boldsymbol{y}]$ we get

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

Then we need to find an expression for $E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2]$. And we do that by adding and subtracting $E[\tilde{\boldsymbol{y}}]$

$$
\begin{aligned}
&E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2] = E[(\boldsymbol{f}-\tilde{\boldsymbol{y}}+E[\tilde{\boldsymbol{y}}]-E[\tilde{\boldsymbol{y}}])^2] \\
&E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2] = E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])^2]+E[(E[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2], \qquad 2E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])(E[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})]=0 \\
&E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2] = E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])^2]+E[(\tilde{\boldsymbol{y}}-E[\tilde{\boldsymbol{y}}])^2] \\
&E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2] = E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])^2]+Var(\tilde{\boldsymbol{y}})
\end{aligned}
$$

Then we put that expression back in above.

$$
\begin{aligned}
E[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= \sigma_\epsilon^2+E[(\boldsymbol{f}-\tilde{\boldsymbol{y}})^2] \\
E[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= E[(\boldsymbol{f}-E[\tilde{\boldsymbol{y}}])^2]+Var(\tilde{\boldsymbol{y}})+\sigma_\epsilon^2 \\
\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \frac{1}{n}\sum_i(f_i-\mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2+\frac{1}{n}\sum_i(\tilde{y}_i-\mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2+\sigma^2.
\end{aligned}
$$