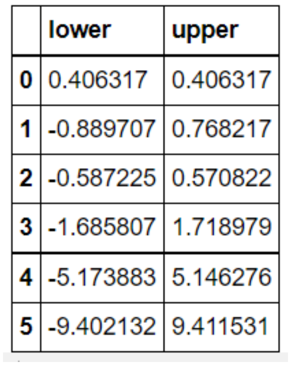
Project 1 report

Exercise 1

In this exercise we generated a set of data by using the Franke function. We then added normally distributed noise to the data and split them into training and test data. We performed a least square analysis with polynomials in *x* and *y* up to fifth order. We tried to scale the data, but it had little impact on the results. The reason for this could be that the values of x and y were between 0 and 1. The difference between the values of the features in the design matrix would therefore be limited.

Table 1 shows the confidence intervals of the parameters β. It seems that the higher the degree of the polynomial is, the bigger the confidence interval is.



	lower	upper
0	0.406317	0.406317
1	-0.889707	0.768217
2	-0.587225	0.570822
3	-1.685807	1.718979
4	-5.173883	5.146276
5	-9.402132	9.411531

table 1: confidence intervals of parameters β

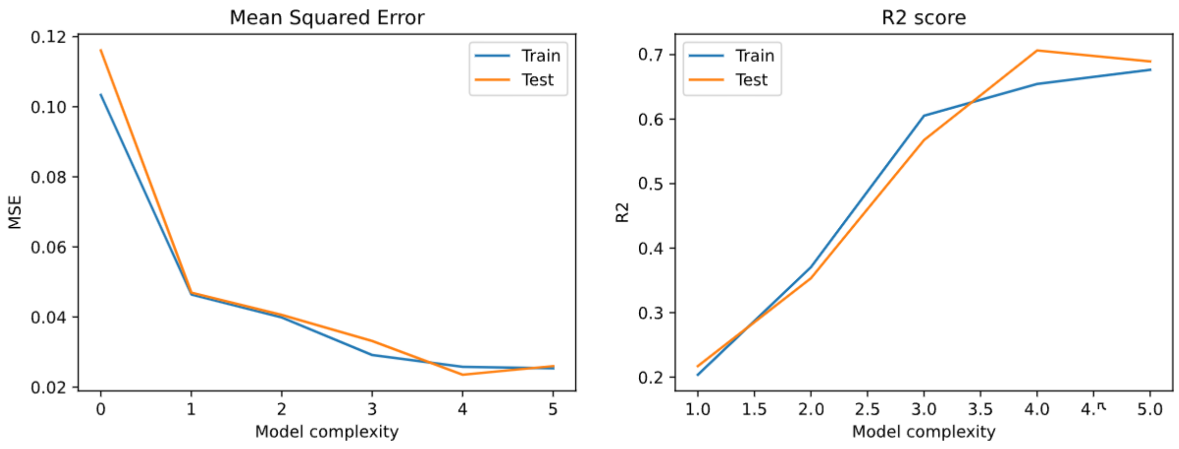


Figure 1.1 & 1.2: MSE and R² with on the x-axis increasing an complexity of polynomial

Figure 1.1 and figure 1.2 show the mean squared error and R² respectively. Both are functions of the polynomial degree. We have used polynomial degrees up to five. The MSE decreases when the polynomial degree increases for those degrees which are used. R² increases when the polynomial degree increases. According to Wikipedia R² is the proportion of the variation in the dependent variable that is predictable from the independent variable(s). Hence the two graphs are consistent with each other.

Exercise 2

The purpose of this exercise was to study the bias-variance trade-off by using bootstrap resampling.

We first made a figure that shows the training and test mean squared error as a function of model complexity. This figure is similar to figure 2.11 in Hastie et al. On the left side of the graph the test curve is close to the training curve, but the MSE is high. Here the bias is high and the variance is low. In the middle of the graph the test curve is low, and both the bias and variance are low. On the right side of the graph the test curve rises again. Here the bias is low and the variance is high.

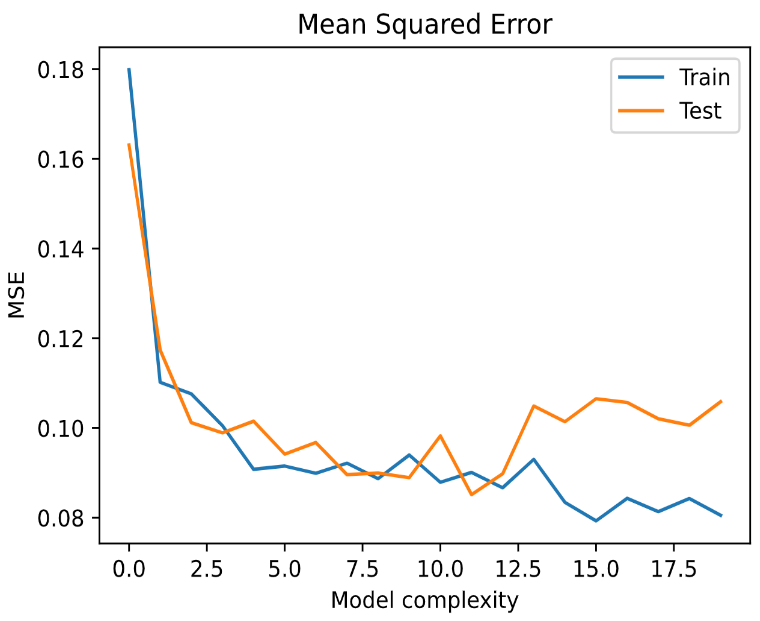


Figure 2.1: This figure is made trying to replicate figure 2.11 of Hastie. Made with the code in ex1.ipynb with 50 data points, 0.3 noise and random state 40.

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

$E[\epsilon]=0$, which leaves us with

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

The first term is the bias; the second term is the variance. Bias means systematic error. As mentioned above, the bias is high when the polynomial degree is low. The model will not fit the data well. This applies both to the training data and the test data. When the polynomial degree is too high, one gets overfitting. The model will not fit the test data well. The variance is high.

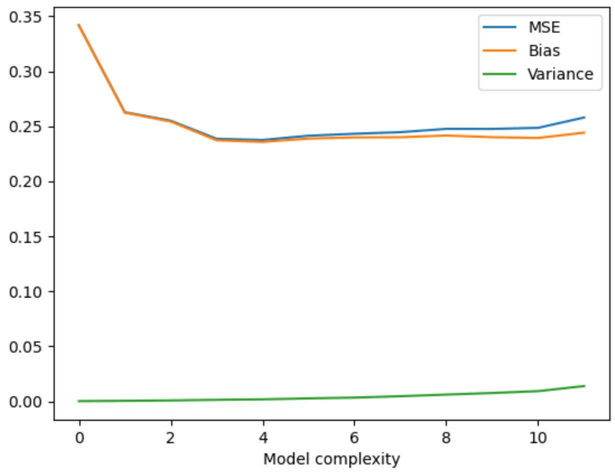


Figure 2.2: bias-variance tradeoff with on the x-axis an increasing complexity of polynomial

We performed a bias-variance analysis of the Franke function by using the bootstrap resampling method. As figure 2.2 shows, the bias is high when the model complexity is low, and the variance increases when the degree of the polynomial becomes high. The mean squared error is approximately the sum of the bias and the variance.

Exercise 3

In exercise 3 we wrote our own code and used the cross-validation technique to analyse our data. We implemented k-fold cross-validation, which means the sample gets split into k different equal sized sub samples. One of the k subsamples will function as test and the others as training data. the cross-validation process is then repeated k times, with having all of the subsamples once as test data
Figure 3.1 displays the mean of a 10-fold cross-validation analysis. In the same figure we also displayed the same test done with the code from Scikit Learn. As you can see both codes will result in the same MSE-function. The MSE rises a little when the complexity of the model rises too. This is because of the increasing variance

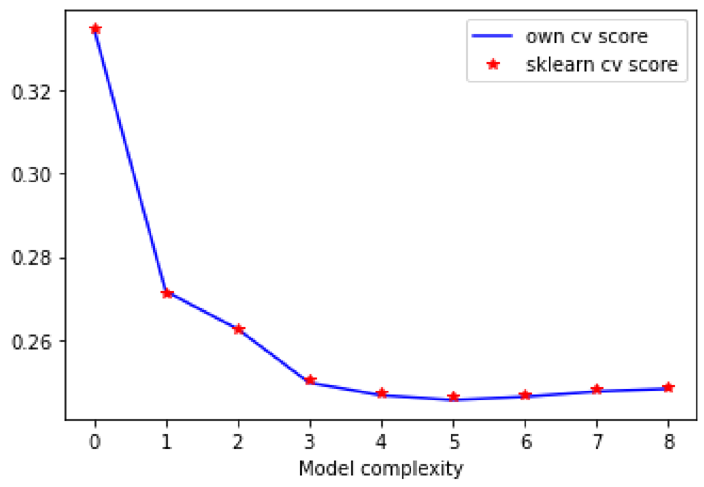


Figure 3.1: Mean squared error with cross validation technique(own code) compared to the cross validation from sklearn

In figure 3.2, 3.3 & 3.4 we displayed a comparison of the MSE between the cross-validation and the bootstrap method. As the complexity of the model rises, both the bootstrap and cross validation shows an increasing MSE score. This can be explained by the increased variance in the model with higher degrees of the polynomial. The complicated model will fit the training data better, but might lead to overfitting as seen with both bootstrap and cross validation. This is also consistent with what we saw in figure 2.2 with the increasing variance.

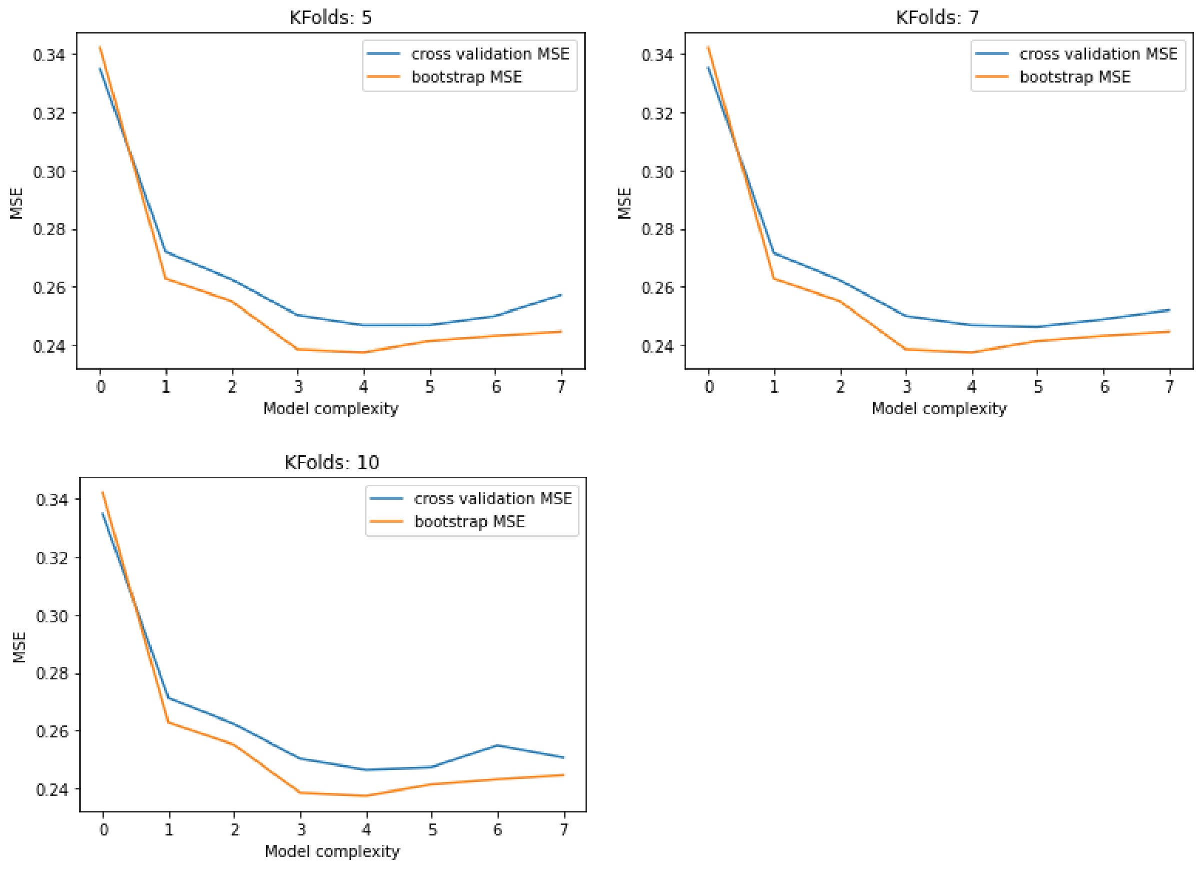


Figure 3.2, 3.3 & 3.4: cross validation compared with bootstrap with different k-folds and on the x-axis an increasing complexity of polynomial

Exercise 4

For this exercise we used Ridge method for our models. We used the same bootstrap from Ex.2 and the same cross validation from Ex.3, but plotted them against different values for lambda.

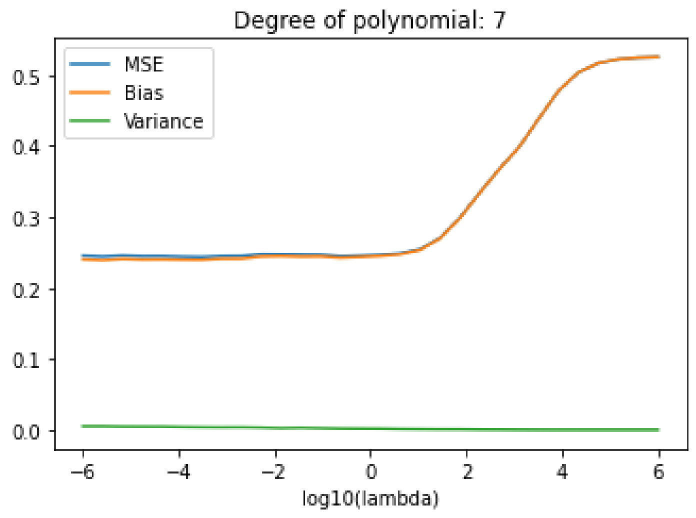


Figure 4.1: bias-variance tradeoff with on the x-axis an increasing lambda in a logscale. Method used: Ridge

In figure 4.1 you will see that the variance stays the same for every value of lambda, but for lambdas from around 10^{1} and higher, the bias will start increasing. Because the variance is that small and doesn't change, the MSE is almost totally dependent on the bias. Furthermore if you compare figure 4.1 to figure 2.2 from exercise 2 you will see that the ridge method did not improve the model. None of the values of lambda created a lower value of MSE than the linear model from exercise 2. From this we can conclude that the ridge model was not the best model to use for our datasets. Maybe if we would use other datasets, we could see an improvement with the ridge regression.

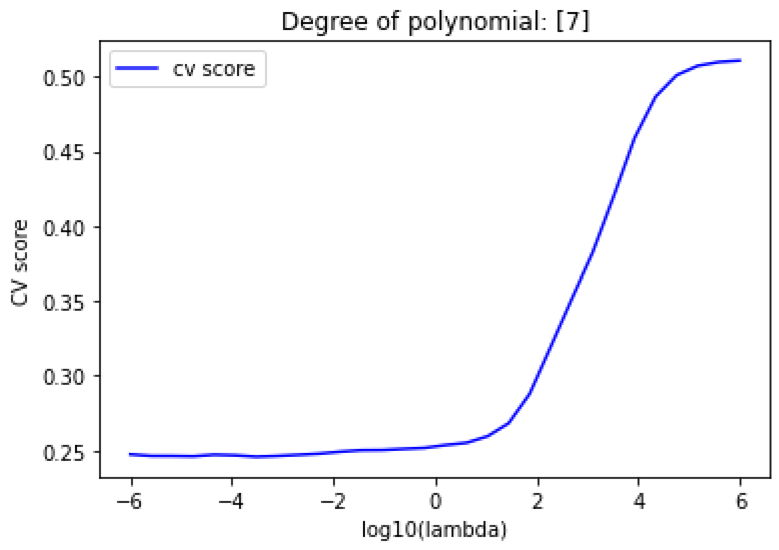


Figure 4.2: Mean squared error with a degree of polynomial of 7 against an increasing lambda in a logscale

To make a comparison to a ridge model, we took the minimal CV score using a grid search to find the optimal parameters for degree of polynomial and lambda. This can be seen in figures 4.2 and 4.3 which shows the optimal parameters versus model complexity and the different lambda values.

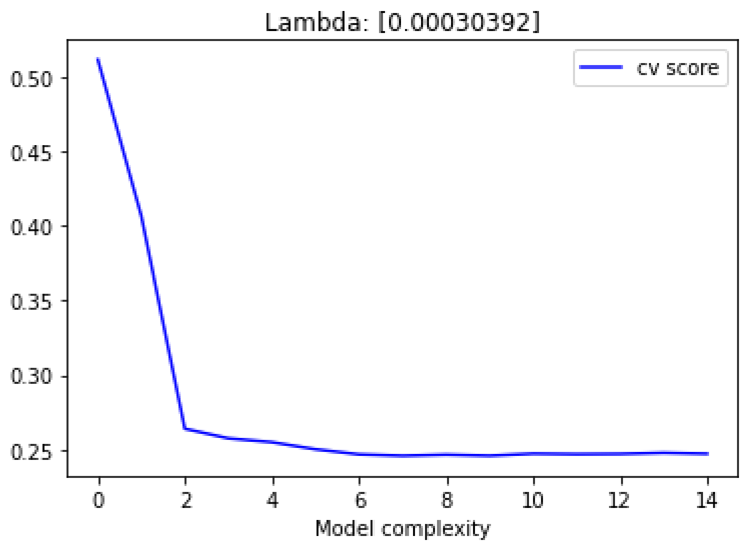


Figure 4.3: Mean squared error with cross validation technique against an increasing model complexity. degree of polynomial: 7. True lambda: 0.0003

With this lambda we recreated figure 3.1, but then with the Ridge model. As can be seen the ridge model now does not rise nearly as much as the linear model does. This way the ridge technique improved the model for higher complexities a lot. Ridge did however not improve MSE value for the lower complexities so they both remain equally good models for both linear and ridge regressions.

In figure 4.4 we can now see that the cross validation MSE and the bootstrap MSE only have small differences if we use different values for lambda. The bootstrap MSE seems to do a little better for lower values of lambda, while the cross validation MSE seems to do a little better for higher values of lambda.

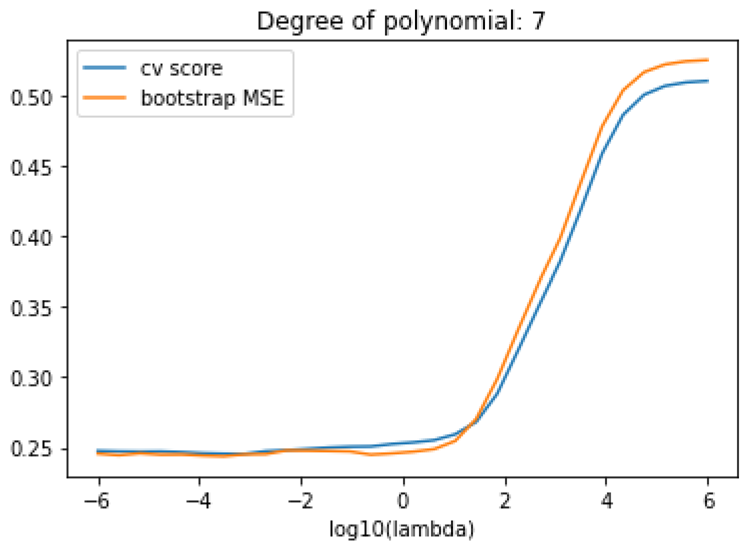


Figure 4.4: MSE bootstrap vs MSE cross-validation with an increasing Log10(lambda) on the x-axis. Method used: Ridge

Exercise 5

The goal of this exercise is to use Lasso regression to fit some data and to see how good this model is compared to other regression techniques. Figure 5.1 displays a bias-variance tradeoff against an increasing lambda. One can see that the variance is almost zero for every value of lambda. For that reason the MSE depends entirely on the value of the bias. The MSE increases until the value of lambda reaches about 1/10. After that it remains constant or almost constant.

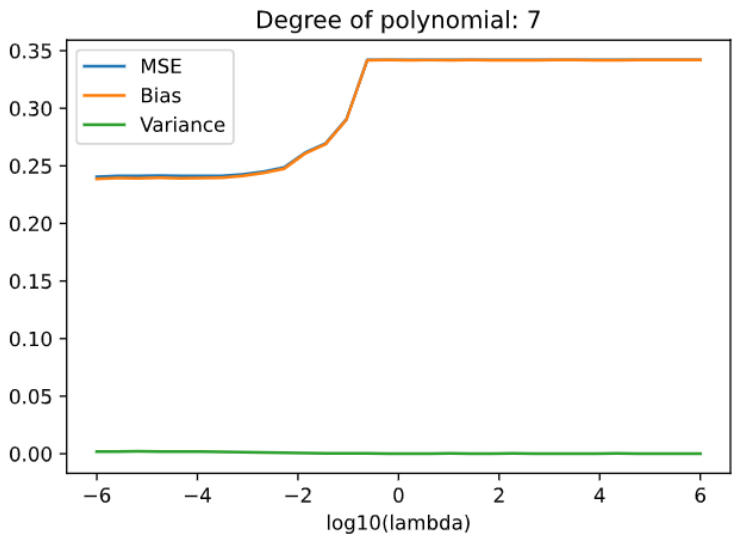


Figure 5.1: bias-variance tradeoff with on the x-axis an increasing lambda in a logarithmic scale. Method used: Lasso

Figure 5.2 displays a comparison of the bootstrap MSE to the cross-validation MSE. The two curves are quite similar, but for all values of lambda the bootstrap MSE is slightly lower than the cross-validation MSE.

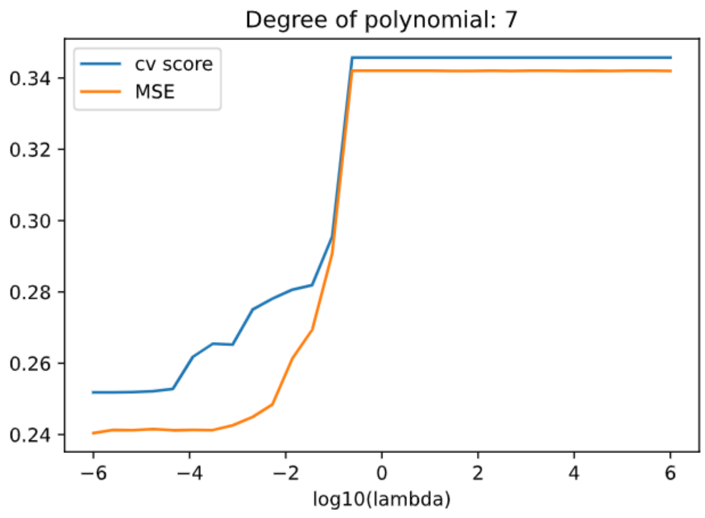
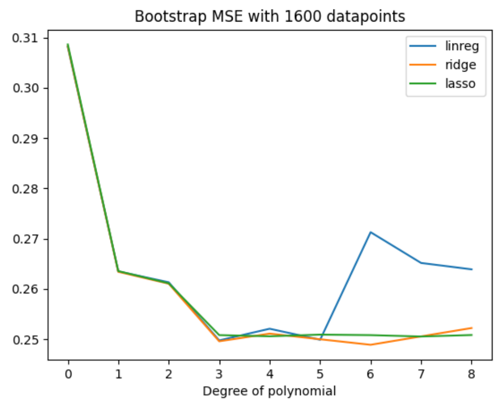


Figure 5.2: MSE bootstrap vs MSE cross-validation against an increasing Log10(lambda) on the x-axis. Regression technique used: Lasso

We then compared Lasso regression with Ridge-and OLS regression. To make a good comparison we've put the three regression models in one graph. For figure 5.3 & 5.4 we did this with bootstrap, but to make a fair comparison we did the same in figure 5.5 & 5.6 with cross-validation. Figure 5.3 displays this with 1600 data points and figure 5.4 displays this with 10000 data points.
As you can see in both graphs there's almost no difference for lower complexities. In figure 5.3 we can however see that when the complexity of the polynomial increases the three models start to perform differently. Lasso and Ridge perform about the same, while the linear regression has the highest score here. These differences probably occur because of the limited amount of data points.
In figure 5.4 this does not happen. Ridge and Linear regression will perform almost the same for all complexities and Lasso will give a slightly higher MSE for higher complexities.



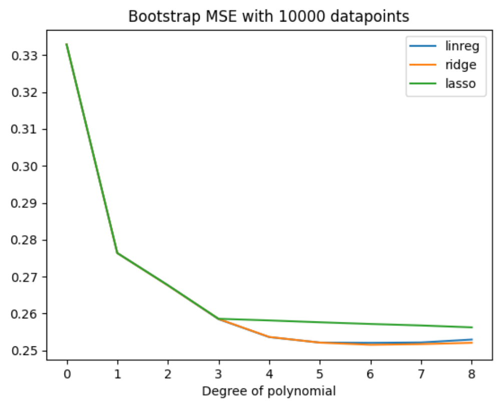
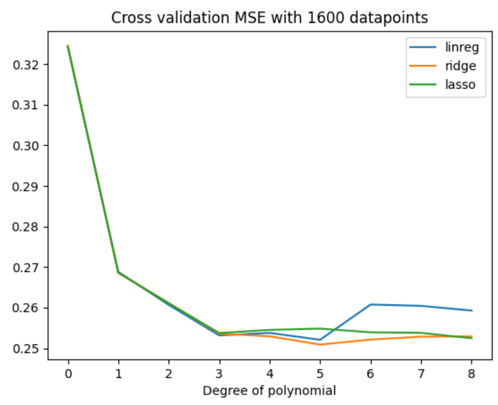


Figure 5.3 & 5.4: MSE as a function of model complexity compared for linear, ridge and lasso regression. Resampling technique: bootstrap. fig 5.3 displays this for 1600 datapoints and fig 5.4 displays this for 10000 datapoints

Subsequently we created the same two graphs with cross-validation as a resampling technique. One can see that with a rising complexity in figure 5.5, the MSE for linear regression will increase a lot. This will probably be the result of an increasing variance. The shrinking parameter introduced in Lasso and Ridge will counteract this variance by smoothing the function to reduce overfitting the training data. Figure 5.6 is very similar to figure 5.3, with Lasso and Ridge giving the best fit for this model. From this we can conclude that it is very dependent on the type of data that you have, which regression method you can use best for your data analysis. This will for instance depend on the amount of data that is available or maybe other variables. Furthermore, you will have to find out for each dataset which regression and resampling technique will give the best result for your data.



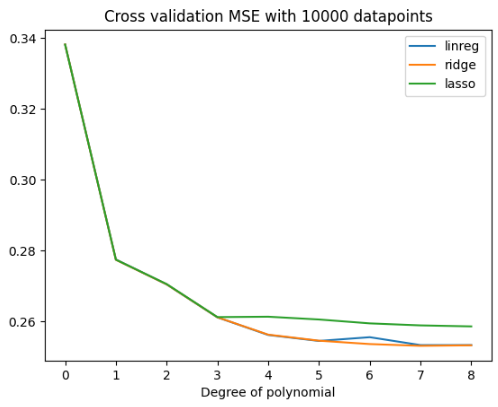


Figure 5.5 & 5.6: MSE as a function of model complexity compared for linear, ridge and lasso regression. Resampling technique: cross-validation. fig 5.5 displays this for 1600 datapoints and fig 5.6 displays this for 10000 datapoints

Exercise 6

For exercise 6 we used terrain data from Norway to test the three regression techniques we've been using in the previous exercises. Cross-validation has been used as a resampling technique for evaluating the model. Figure 6.1 displays the data we used with a reduced amount of data point(pixels). We did that to get more interesting results comparing the three regression techniques. With the amount of data we could have used, every test point would for example be surrounded by a high amount of train data. We wanted the data to resemble a situation that would not be that ideal, so we reduced the amount of data points.

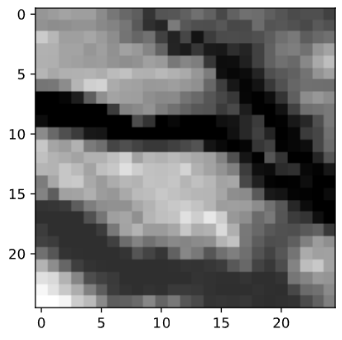


Figure 6.1: Terrain data used to make the models in this exercise

Ordinary Least Squares

First we created a graph for the MSE cross-validation score to see which degree of polynomial would give the best fit for the data. In figure 6.2 we can see that this would be around a degree of polynomial of 11 or 12. If the degree increases more, the model begins to overfit due to an increasing variance. Figure 6.3 displays the R2 score. This R2 score confirms that the ideal degree of polynomial would be around 11 or 12.

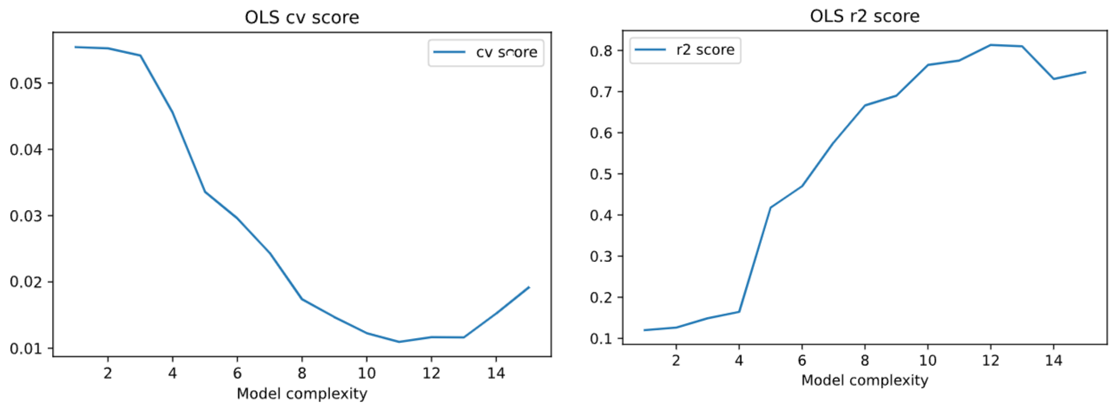


Figure 6.2 & 6.3: Cross-validation MSE and R2 score against an increasing degree of polynomial for OLS

Subsequently we made a model that tries to predict the original image with the OLS technique. For this we used a 12th degree polynomial. One can see in figure 6.4 that the image the model created resembles the original image quite a lot.

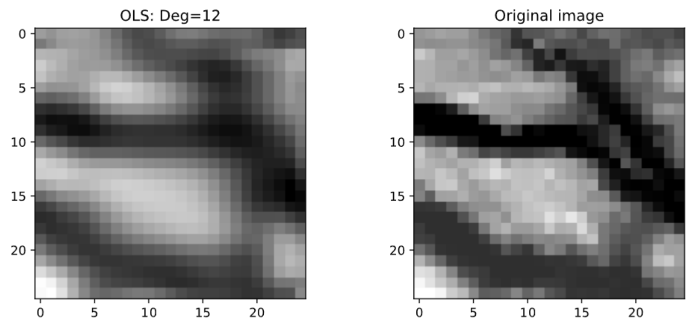
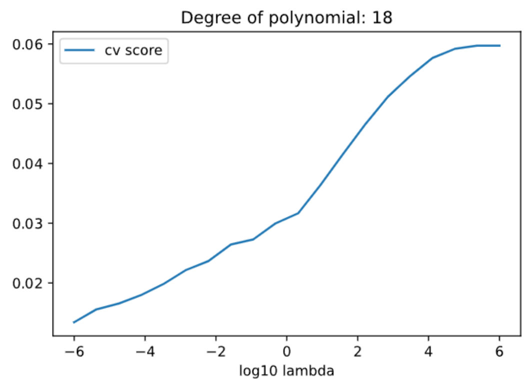


Figure 6.4: Prediction of the original image with OLS

Ridge

Next we created the cross-validation MSE score graph with Ridge regression and pinned the degree of polynomial at 18. In Figure 6.5 one can see the cross-validation MSE score against lambda. It becomes clear that the lower lambda is, the lower the MSE score will be. This suggests that Lambda being zero would give the best fit, which is essentially the same as OLS. With lambda being zero there will be no shrinking of the model parameters, which in turn will make the model equal to OLS.



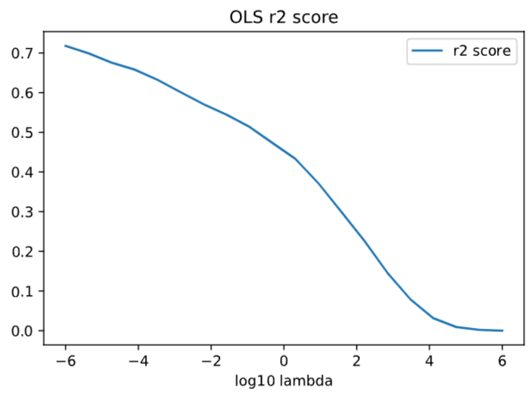


Figure 6.5 & 6.6: Cross-validation MSE and R2 score against an increasing degree of polynomial for Ridge

Looking at the Graph for the R2 squared confirms this again as the lowest value of lambda will give the highest R2 score. This can be seen in figure 6.6

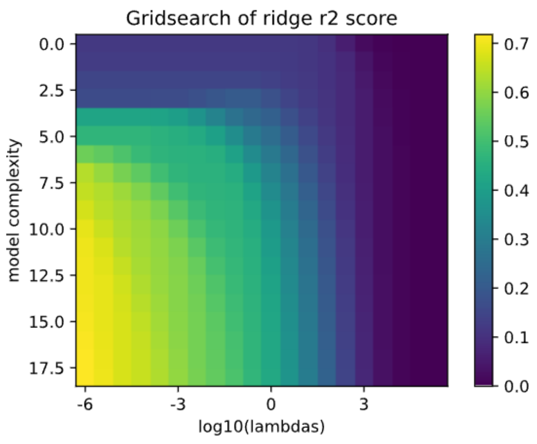


Figure 6.7: Gridgraph of R2 score plotted against model complexity and lambda for Ridge

Figure 6.7 displays a gridsearch where the R2-score is plotted against the complexity of the model and the value of lambda. On the colors of the grid one can see that the lowest value of lambda, which corresponds to the OLS model, performs the best on the data. This again confirms all the previous assumptions

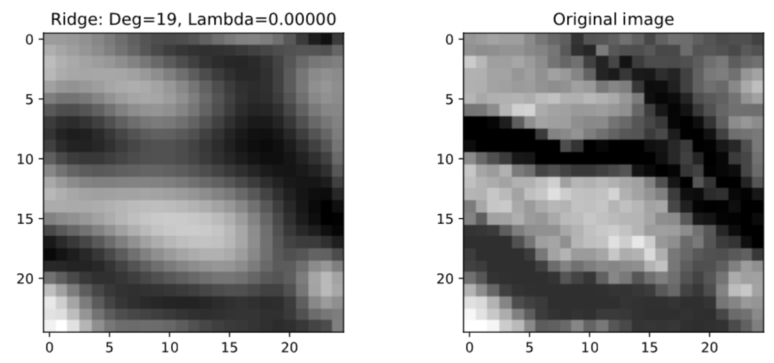


Figure 6.8: Prediction of the original image with Ridge

The model now predicts an image that still resembles the original image quite well. you would think that a degree of polynomial of 11 or 12 would be the best since this was the case for OLS, but because lambda is not perfectly zero, it seems that a degree of 18 will give the best results in this case.

Lasso

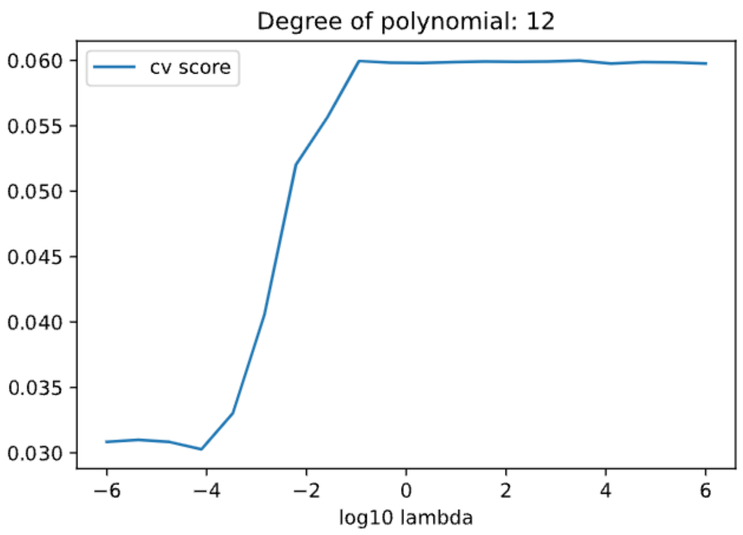


Figure 6.9. Cross validation MSE score as function of log10 lambda

Figure 6.9 shows that when we use cross validation as a resampling technique, the MSE score is low when the value of lambda is low. The graph resembles the corresponding graph for Ridge regression. When lambda is low, Lasso regression will give approximately the same results as ordinary least squares regression.

Figure 6.10 shows a gridsearch for R² score using Lasso regression. The figure is quite similar to the corresponding figure for Ridge regression. The R² score is high when lambda is low. This indicates that ordinary least squares regression gives better results than Lasso regression. One can also see that the highest R² score is higher for Ridge regression than for Lasso regression.

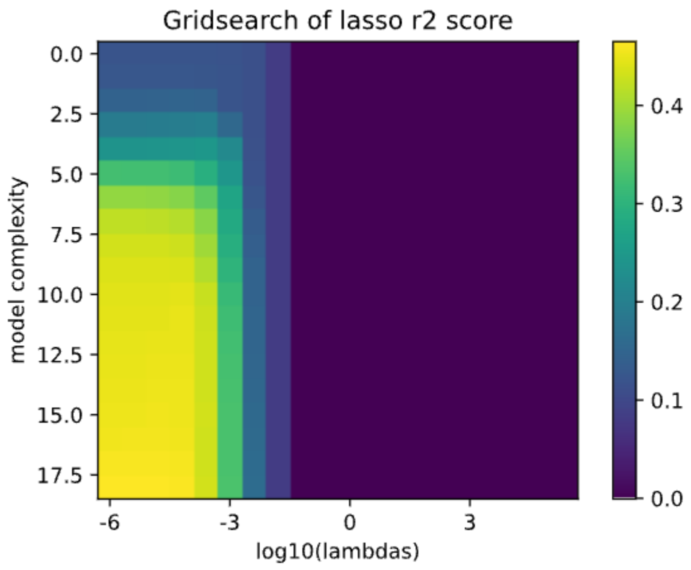


Figure 6.10. Gridsearch for R² score using Lasso regression. R² score is plotted as a function of model complexity and log10 lambda.

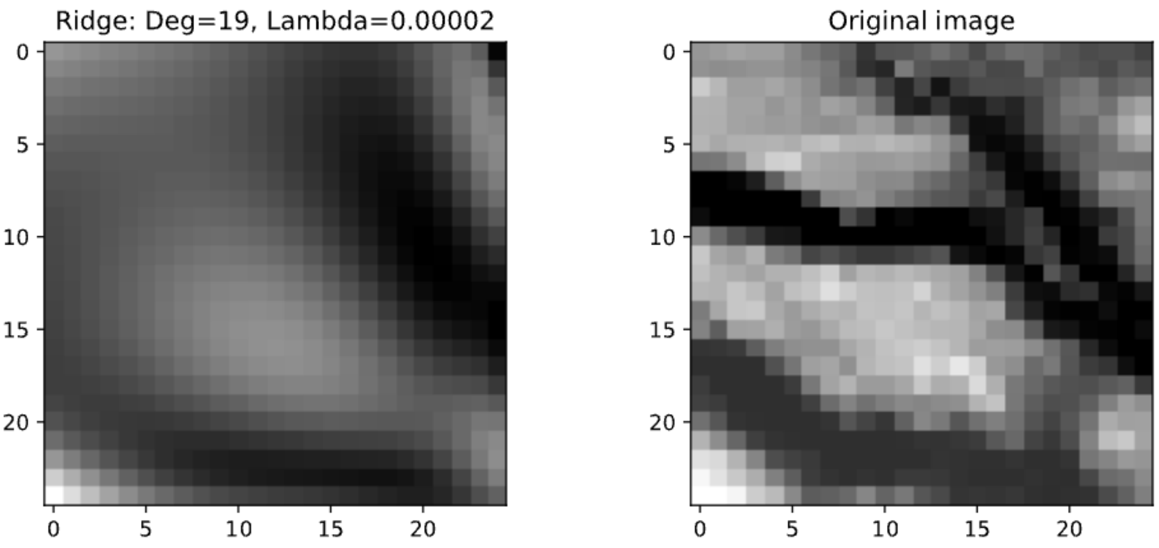


Figure 6.11. The original image of the terrain (right) and the model (left). Lasso regression.

Figure 6.11 shows the original image of the terrain (right) and the model (left). The recreation (the model) of the terrain is not as good as for Ridge regression or ordinary least squares regression. This can be attributed to the way Lasso minimizes its cost function. In theory, Lasso with a lambda value of approximately zero will give the same result as regular OLS. This is also something we saw with Ridge when the shrinking parameter was close to zero. The difference comes from the minimization strategy of these functions. OLS and Ridge both have analytical solutions when trying to find the optimal parameters. Lasso needs to use some form of gradient descent techniques to find the global minimum. In our case it seems that the model got stuck in a local minimum, leading to a much worse prediction.

Conclusion

Our regression methods reproduce the part of the terrain that we have chosen, fairly well. The fact that the R^2 score is high when lambda is low for Ridge and Lasso regression, indicates that ordinary least squares regression gives better results than Ridge and Lasso regression when applied to our data. Comparing Lasso to the rest might be unfair considering the global minimum was not found, but as seen from the previous exercises where we predict the Franke landscape, Ridge was often on par with or better than Lasso in most cases.