FYS-STK4155: Project 1

Bjørn Inge Vik[a], [b]

[a] *bingvik@gmail.com*
[b] *https://github.com/bingovik/FYS-STK4155*$_P$1

Abstract: I explore three different linear regression methods (OLS, Ridge and Lasso) applied to generated data by Franke's function and on real terrain data for a location in Norway. The regressions were applied to polynomials of coordinates in two dimensions. The analysis finds that all three models, with a relatively wide range of polynomial orders, should work reasonably well to estimate missing data points. The values of Franke's function in the domain used are between 0 and 1 and the best MSE achieved was on the order of 0.01. Terrain data altitudes in the area of investigation was between 0 and 24 meters and the best achieved MSE was roughly 10 meters.

1. Introduction

The popularity and applicability of machine learning have soared in recent years, thanks to availablitiy of data as well as improvements in algorithms and computational power. Linear regression methods may be viewed as simple and transparent examples of machine learning algorithms.

In this report, I will investigate three different linear regression methods applied on generated data by Franke's function and on real terrain data for a location in Norway. Franke's function is a widely used test function for evaluating interpolation and fitting algorithms.

The goal of this project is to find the model that is best suited to describe the data. Also, I will include some results intended to shine light on the theoretical background for evaluating and comparing different models.

The report has three main sections:

- Methods and data: Description of the input data and the models tested. This part also includes some theoretical background and analytical things.
- Analysis: Testing and analysis of model results.
- Conclusion: Summary and discussion of results.

2. Data and Methods

2.1. Data

2.1.1. Franke's Function

Franke's function is given by

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

It has been evaluated on a 40x40 grid of x and y with coordinates picked randomly from a uniform distribution $[0,1)$. Gaussian random noise with standard deviation $\sigma = 0.1$ was added on top. A plot of the data is shown in figure 1.

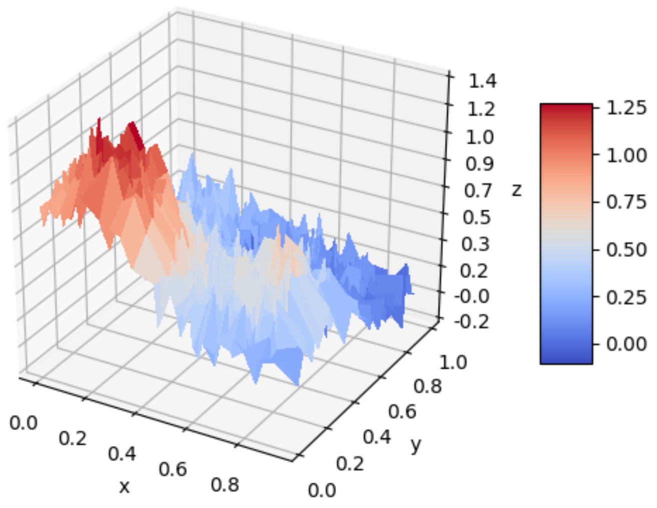


Figure 1: Franke's function with added noise with standard deviation 0.1 evaluated on a randomly generated 40 by 40 grid.

2.1.2. Terrain Data

I have analyzed terrain data for a location in Norway. I have used the image file *SRTM_data_Norway_2.tif* which was provided by the instructors for this project. To save computational time, in particular for Lasso regression, only the top right corner of the image was used (containing 36x18 data points). Figure 2 and 3 show the terrain data.

2.2. Methods

Three types of regression were tested using polynomials of different degrees created from the x and y coordinates of the data sets: Ordinary linear regression (OLS), Ridge regression and Lasso regression. These methods are described in this section. I also describe the metrics used and theoretical background for bias and variance considerations.

2.2.1. Ordinary Least Squares Regression

Ordinary least squares (OLS) regression seeks to estimate a target variable y using a linear combination of a set of p explanatory variables x_j:

$$\tilde{y} = \beta_0 + \sum_{j=1}^{p} \beta_i x_i$$

The goal is to minimize the mean squared error of the prediction over n data points in the sample:

$$f_{cost} = MSE = \frac{1}{N} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{1}$$

by tuning the coefficients β_j. By listing the data points along the rows in vectors, equation 1 can be written in matrix form as

$$f_{cost} = MSE = \frac{1}{N} (\mathbf{y} - \tilde{\mathbf{y}})^T (\mathbf{y} - \tilde{\mathbf{y}})$$

There is an exact analytical solution for the optimal coefficients $\boldsymbol{\beta}$ that minimizes the cost function. By defining a design matrix $\boldsymbol{X}$ containing n samples along the rows and p explanatory variables plus a bias unit along the columns the solution can be written

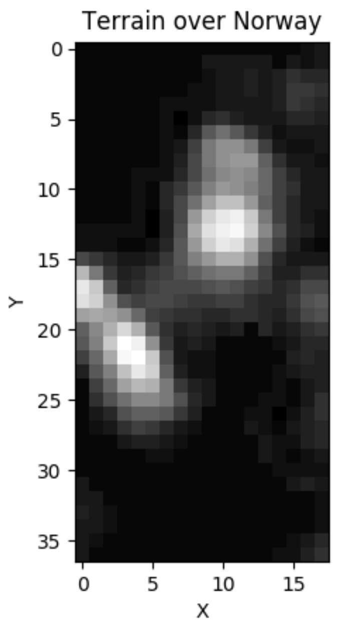


Figure 2: Terrain around Møsvatn, Austfjell (gray scale)

$$\boldsymbol{\beta} = \left(\mathbf{X^T X}\right)^{\mathbf{-1}} \mathbf{X^T y} \tag{2}$$

I have implemented and used the analytical solution (see Appendix Appendix A). As an alternative I have used Scikit-Learn's *Linear Regression Model*.

2.2.2. Ridge Regression

Ridge regression finds the optimal parameters $\boldsymbol{\beta}$ by minimizing the modified cost function:

$$f_{cost} = \frac{1}{N}\left(\mathbf{y} - \tilde{\mathbf{y}}\right)^T \left(\mathbf{y} - \tilde{\mathbf{y}}\right) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}$$

Where a so-called L2 regularization term has been added to the corresponding equation for OLS. The regularization parameter λ punishes the coefficients, with the cost increasing quadratically. This is an effective way of combating overfitting. This can be a serious problem if the number of explanatory variables is relatively high compared to the number of data points or if there is a lot of noise or outliers in the data.

The regularization factor lambda is a hyperparameter in the model that is typically set by trial and error. A too low value can result in the model performing significantly worse on test data than on training data. A too high value will result in the model not capturing sufficient detail in the data. A value of zero recovers the cost function and solutions for OLS regression.

The exact analytical solution for the coefficients in Ridge regression is

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

Another advantage of Ridge regression is that the matrix in brackets is always invertible. I have implemented the above equation in a function and I have also used Scikit-Learns *Ridge* model.

2.2.3. Lasso Regression

Lasso stands for least absolute shrinkage and selector operator. The cost function used is

$$f_{cost} = \frac{1}{N}\left(\mathbf{y} - \tilde{\mathbf{y}}\right)^T \left(\mathbf{y} - \tilde{\mathbf{y}}\right) + \lambda \sum_{j=0}^{p} \beta_j$$

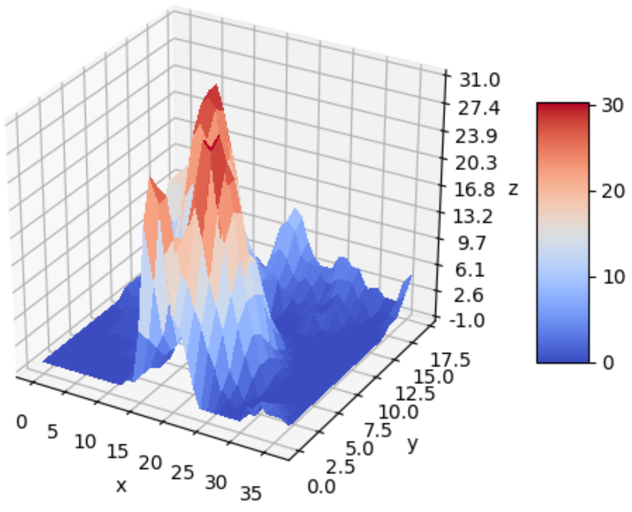


Figure 3: Terrain around Møsvatn, Austfjell (surface plot)

It is similar to the expression for Ridge, but the first order regularization of the coefficients has two important consequences. First, there is no analytical solution that minimizes the cost function; the optimal coefficients have be determined numerically. Second, the regularisation can cause some of the coefficients to become zero, as opposed to in Ridge regression where they would be reduced but stay non-zero. Lasso can thus be used as a feature reduction algorithm. I have used Scikit-Learn's *Lasso* model to test this type of regression.

2.2.4. Bias Variance Trade-off

The MSE can be decomposed into a bias term, a variance term and an irreducible noise term. The following derivation on Wikipedia[1] takes f to be the true underlying function, y to be the observed values and $\tilde{y}$ to be the model prediction.

$$MSE = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]$$

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

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

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

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

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

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

$$= bias[\tilde{\boldsymbol{y}}]^2 + var[\boldsymbol{y}] + var[\tilde{\boldsymbol{y}}]$$

$$= bias[\tilde{\boldsymbol{y}}]^2 + \sigma^2 + var[\tilde{\boldsymbol{y}}]$$

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

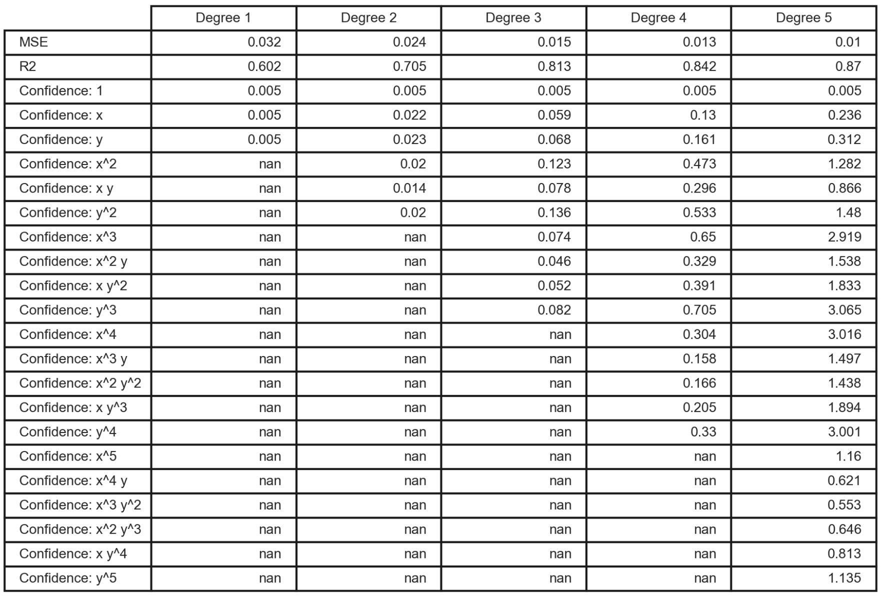
This illustrates that there is a trade off in the model selection. A model with low complexity or with strong regularization may have little variance, i.e. the predictions are expected to vary little if the model is trained again on new data from the same distribution (with the only difference being the random noise component). But such a model is likely to suffer from relatively high bias, i.e. being unable to accurately predict each individual data point. By contrast, a complex model may have low bias and high variance. High variance essentially means that the model is overfitting the training data. As mentioned, the risk of this occurring is greater if the number of explanatory variables is relatively high compared to the number of data points or if there is a lot of noise or outliers in the data.

3. Analysis

3.1. Franke's Function

3.1.1. OLS on Whole Sample

As a first test of the suitability of linear regression to estimate Franke's function I applied the self implemented OLS on the complete data set (without splitting into train and test data). Results for polynomials up to fifth degree are shown in figure 4. As expected, the MSE and R2 score improve with model complexity. The table also shows estimates for the confidence intervals of the parameters. They increase with model complexity, illustrating that higher order models are more unstable. Figure 5 shows a comparison for models up to 13th order of the analytical self-implementation and Scikit-Learns implementation of OLS. The performance is almost identical for polynomials up to and including 11th order, but the error for the self-implemented method blows up for more complex models. This suggests that the matrix inversions performed in 2 becomes numerically unstable. Scikit-Learn's algorithm appears to handle this better.



	Degree 1	Degree 2	Degree 3	Degree 4	Degree 5
MSE	0.032	0.024	0.015	0.013	0.01
R2	0.602	0.705	0.813	0.842	0.87
Confidence: 1	0.005	0.005	0.005	0.005	0.005
Confidence: x	0.005	0.022	0.059	0.13	0.236
Confidence: y	0.005	0.023	0.068	0.161	0.312
Confidence: x^2	nan	0.02	0.123	0.473	1.282
Confidence: x y	nan	0.014	0.078	0.296	0.866
Confidence: y^2	nan	0.02	0.136	0.533	1.48
Confidence: x^3	nan	nan	0.074	0.65	2.919
Confidence: x^2 y	nan	nan	0.046	0.329	1.538
Confidence: x y^2	nan	nan	0.052	0.391	1.833
Confidence: y^3	nan	nan	0.082	0.705	3.065
Confidence: x^4	nan	nan	nan	0.304	3.016
Confidence: x^3 y	nan	nan	nan	0.158	1.497
Confidence: x^2 y^2	nan	nan	nan	0.166	1.438
Confidence: x y^3	nan	nan	nan	0.205	1.894
Confidence: y^4	nan	nan	nan	0.33	3.001
Confidence: x^5	nan	nan	nan	nan	1.16
Confidence: x^4 y	nan	nan	nan	nan	0.621
Confidence: x^3 y^2	nan	nan	nan	nan	0.553
Confidence: x^2 y^3	nan	nan	nan	nan	0.646
Confidence: x y^4	nan	nan	nan	nan	0.813
Confidence: y^5	nan	nan	nan	nan	1.135

Figure 4: OLS regression using the whole data set. The table shows the metrics MSE and R2 as well as an estimate for the 95%confidence interval (assuming normal distribution) of the parameters β for each explanatory variable in the polynomial. Models with polynomials up to fifth degree are shown.

3.1.2. OLS evaluation

Using Scikit-Learn's function *train_test_split*, the data set was shuffled and randomly split into a training sample consisting of 80% of the data and the rest reserved as test data. Figure 6 shows the mean squared error evaluated on the training data and test data for polynomials up to degree 20 using Scikit-Learn. Performance on the test set starts to deviate and become slightly unstable for degree eight and higher. The fact that the performance gains for polynomials higher than degree eight on the training set is not mirrored in the test set is a sign of overfitting. That said, performance on the test set remains rather good for high order polynomials,

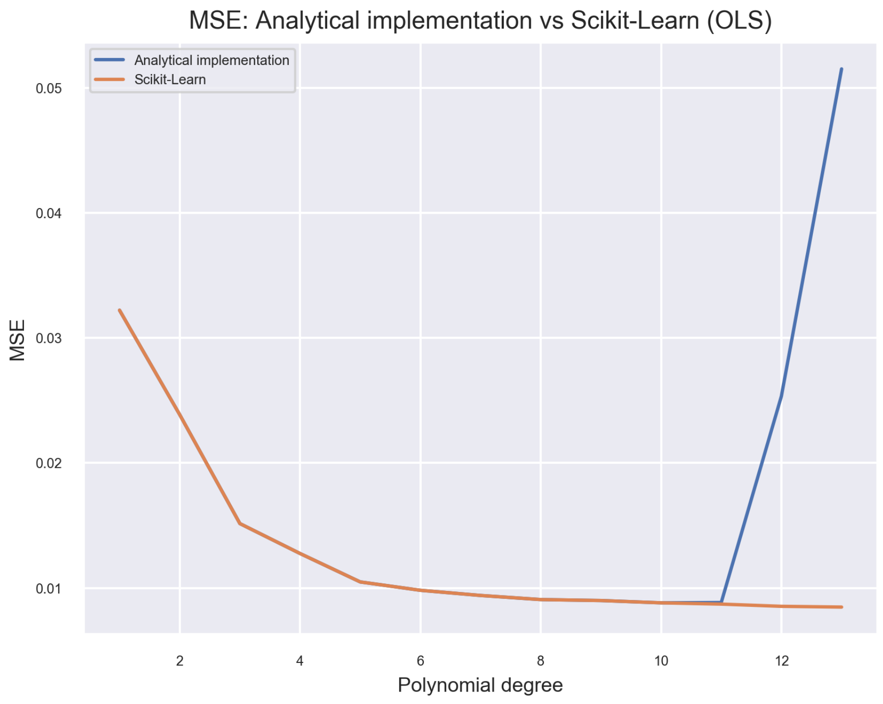


Figure 5: MSE using the self-implemented analytical OLS and using Scikit-Learn's Linear Regression model. Scikit-Learn's algorithm appears numerically more stable for high order models.

albeit somewhat unstable.

I have used k-fold cross validation to evaluate OLS regression models with different polynomial degrees. Cross validation involves dividing the data into a training set and a validation set. The model is trained on the training set and metrics such as the MSE are calculated on the validation set. This is repeated a number of times, each time with a completely different validation set but with partly overlapping training sets. Cross validation should lead to more realistic determinations of the predictive power of the model on new data. The code used to perform k-fold cross validation is included in Appendix C. Figure 7 shows the average MSEs on the validation sets and for the separate test set containing 20% of the data.

A bias-variance analysis was performed using cross validation and bootstrapping. The latter technique involves randomly sampling data points from the data set with replacement until a set with the original size is obtained. Then a regression is fitted to this data and then used to predict values for separate test data. This procedure was repeated 100 number of times to obtain 100 different estimates for the test data. This way one can calculate the bias and variance for the different estimates by simulating new data sets. Using cross validation, results in Figure 8 is showing how the bias drops sharply with model complexity in the beginning, while the variance slowly becomes a factor for higher order models. Figure 9 shows results using bootstrapping. Estimates for variance are higher and more unstable in the example shown, but different runs with the same number of data points and noise level can often produce much smoother results.

3.1.3. Ridge evaluation

I have used k-fold cross validation to evaluate Ridge regression models with different polynomial degrees and regularization lambda. Figure 10 shows the average MSE on the validation for models with different complexity and regularization. The pattern is of better predictions for higher order polynomials and little regularization. This is confirmed when looking at MSE evaluated on the separate test (figure 11). This test set is the same used to evaluate all models (OLS, Ridge and Lasso). It appears that all Ridge models with polynomial degree between 6 and 20 (possibly higher) and regularization smaller than roughly 0.1 are good models that mimicks Franke's function on the particular data set used. The same results are visualized in a plot showed in figure 12. For some polynomials one can see a slight improvement with for some values of the regularization parameter lambda, but in general it appears not important for MSE. Regularization would be more useful for lowering the MSE of models with even higher degree, and in particular if the data set had fewer data points or if the noise

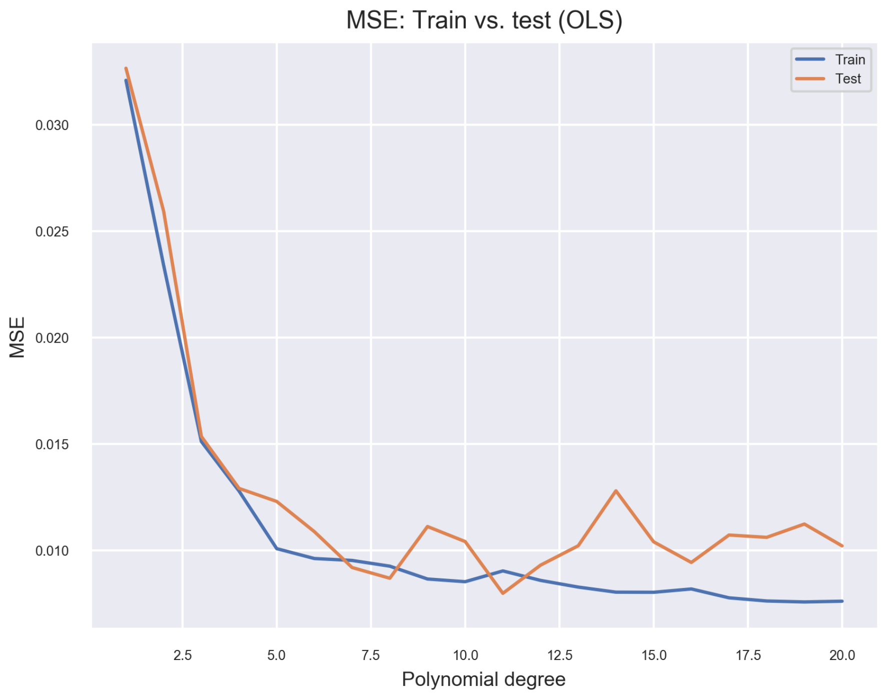


Figure 6: MSE performance on training and test sets using OLS. Overfitting becomes a factor above degree eight or so as improvements on the training set beyond that point is not reflected on the test set.

was cranked up. That said, one can see some effect of regularization in figures 13 and 14, comparing the bias and variance of Ridge models with $\lambda = 10^{-7}$ and $\lambda = 4 * 10^{-3}$ respectively. The more aggressive regularization minimizes variance and increases the bias.

3.1.4. Lasso evaluation

Lasso regression was performed but limited to polynomials up to and including fifth order to save computational time. The MSEs on the validation sets and the test set are shown in figure 15 and 16 respectively. The fifth order models are performing best and clearly it would be interesting to test out higher order models as well. Aggressive regularization leads to poor performance and figure 17 shows how only only the intercept remains non-zero for $\lambda > 0.2$.

3.1.5. Best Models

Best practise is to let the metrics on the validation sets determine the choice of models and then let the performance on the test set be the final judge of the predictive power. OLS regression using a 15th order polynomial recorded an excellent MSE of 0.007 on the validation sets with a corresponding MSE for the test set of 0.011.

Near identical results were achieved for Ridge: a validation score of 0.007 and a test score of 0.011 for an 15th degree polynomial and $\lambda \approx 4 * 10^{-7}$. This is not a surprise as regularization was deemed of little significance for the MSE for the data set used. The Ridge model that performed best on the validations sets gives predictions shown in figure 19.

However, it is also tempting to keep in mind the bias/variance analysis (figures 9 and 13) saying that higher order models are more unstable. In this light, good models for OLS could for example be polynomial degree 7 (MSE roughly 0.01) and either polynomial degree 7 and low λ or a degree 15 with stronger regularization (also MSE roughly 0.01) for Ridge.

For the best validation score, Lasso scored an MSE of 0.012 on the test set (polynomial order five and $\lambda = 10^{-7}$. Figure 18 shows the MSE difference between Ridge and Lasso on the test set (no cross validation). It shows almost identical results for smaller λ. Regularization factors above 0.1 are clearly too aggressive for Lasso. As for Ridge and OLS regression, it is likely that slightly better results could have been achieved for Lasso using

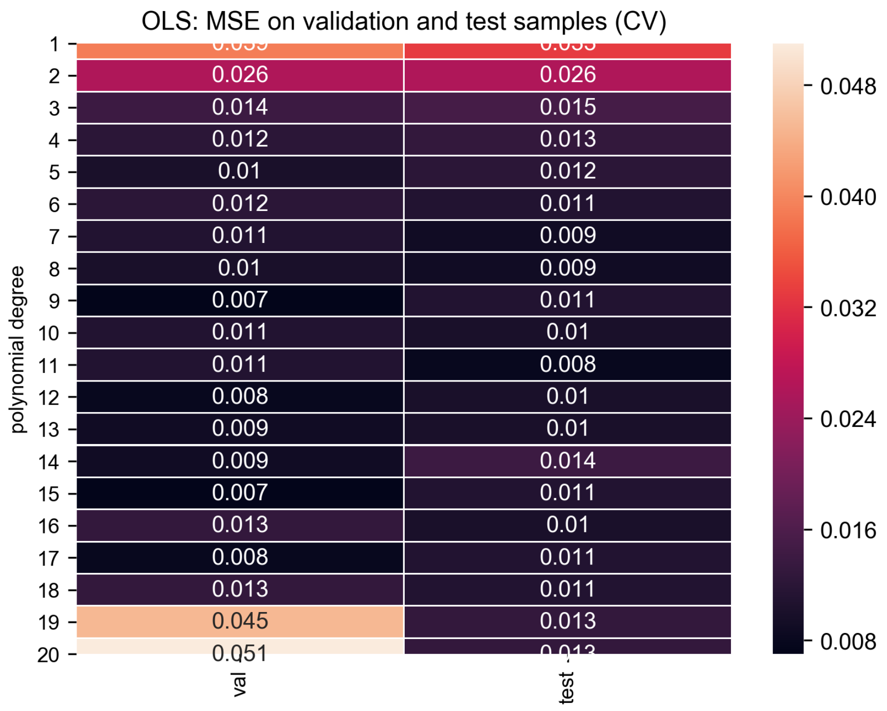


Figure 7: Average MSE for OLS regression calculated on validation sets and the test set using cross validation. Models with polynomial degree 4-18 generally perform well on both sets. There are some signs that higher order models perform worse and are more unstable, at least on the validation sets.

higher order polynomials. But there is nothing to suggest that Lasso would perform better than Ridge or OLS even then.

3.2. Real Terrain Data

3.2.1. OLS evaluation

Cross validation for polynomials up to 20th degree were tested for OLS regression models. Figure 20 shows the average MSEs for the validation sets and for the separate test set containing 20% of the data. Results are unstable due to the relatively small size data set - and especially for the validation sets as they are smaller than the test set. Overall, performance is good for models with polynomial degrees 8-18. Beyond that performance appears to deteriorate and becomes even more unstable.

3.2.2. Ridge evaluation

Cross validation for polynomials up to 20th degree were also tested for Ridge regression models with varying regularization. Figures 21 and 22 show the average mean squared errors for the validation sets and for the separate test set respectively. On the test set, but not on the validation set, deteriorating performance for polynomials of degree 18 and 19 is seen. Model results are likely quite unstable at this high order as the number of explanatory variables (231 for degree 20) approaches the number of data points (532 in the full training sample). There is no clear sign of a regularization effect, which is confirmed by figure 23. The model with polynomial degree 14 shows strong fluctuations with λ, but can not be trusted to be a reliable dependency as it is not mirrored in nearby models.

3.2.3. Lasso evaluation

Lasso regression with cross validation was performed but again limited to polynomials up to fifth order to save computational time. Results for MSE on the test set are shown in figure 25.

3.2.4. Best Models

Again I follow best practise of letting the metrics on the validation sets determine the choice of models and then let the performance on the test set be the final judge of their predictive power.

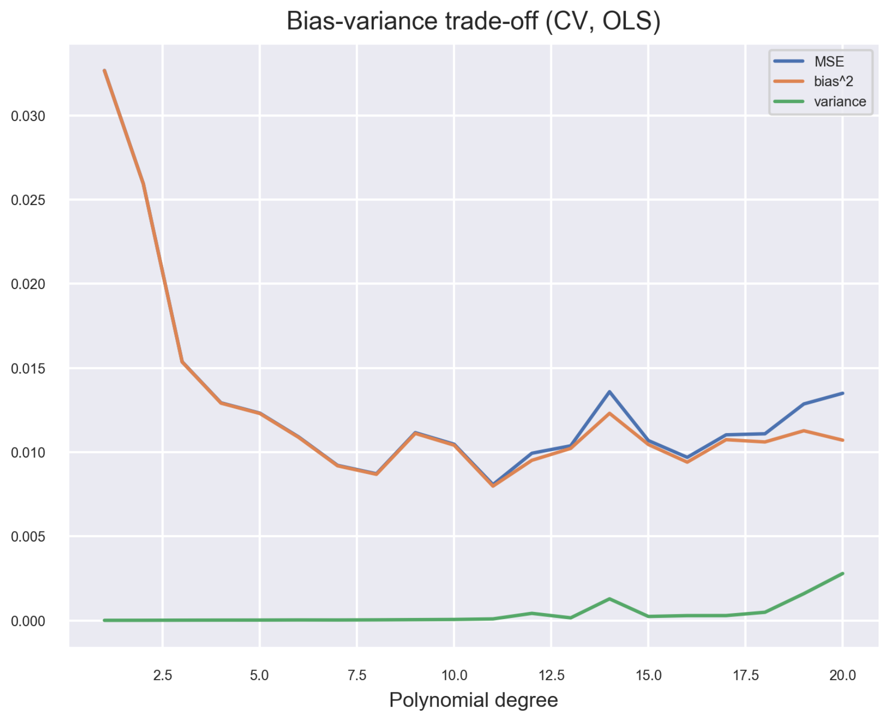


Figure 8: Bias-variance analysis for OLS regression, calculated using cross validation. According to these calculations, model variance, and thus overfitting, becomes a factor for models higher than degree 11 or so.

OLS regression using an 18th order polynomial recorded an extremely low MSE of 2.7 meters on the validation sets with a corresponding MSE for the test set of 9.8 meters. Looking at figure 20, a less rigid approach would be to say that model degree 18 looks too close to the more unstable 19 and 20, and that a safer bet would be to pick a favorite model around degree nine, giving a test MSE of 9.6 meters.

Similar results were achieved for Ridge: a validation score of 4.8 meters and a test score of 9.6 meters for a 16th degree polynomial and $\lambda = 10^{-7}$. This model also sits fairly comfortable in an area of low MSE in figure 21. The Ridge model that performed best on the validations sets gives terrain predictions shown in figure 27.

The best validation score for Lasso was an MSE of 15 for a fourth degree polynomial model with regularization $\lambda = 3.7 * 10^{-3}$. This gave a test MSE of 28 meters. Figure 26 shows the MSE difference between Lasso and Ridge on the test set (no cross validation). Like for Franke's function, it shows almost identical results for smaller λ. Regularization factors above 0.1 are clearly too aggressive for Lasso. But better results could have been achieved for Lasso using higher order polynomials, likely comparable to the results for Ridge regression. But there is nothing to suggest that Lasso would perform better than Ridge or OLS even then.

4. Conclusion

The analysis in this article shows that a wide range of polynomial linear regression methods works reasonably well to estimate both Franke's function and the terrain data used. But it has to be imagined that the challenge is to estimate "missing pixels" in a terrain image or blank values for Franke's function within otherwise well covered areas. The methods described in this article would fail miserably if used to predict data for an area "outside" the area covered by the training data. This is obviously the case for the real terrain data, as the world's geology differs from location to location and is not easily modeled this way.

Alternative ways of estimating missing or faulty data points within a sample would be interpolation methods.

4.1. Franke's Function

The simplest and best models to estimate the data used for Franke's function was really OLS on polynomials somewhere between five and 15. Ridge models with low regularization for the same degrees was also good. For

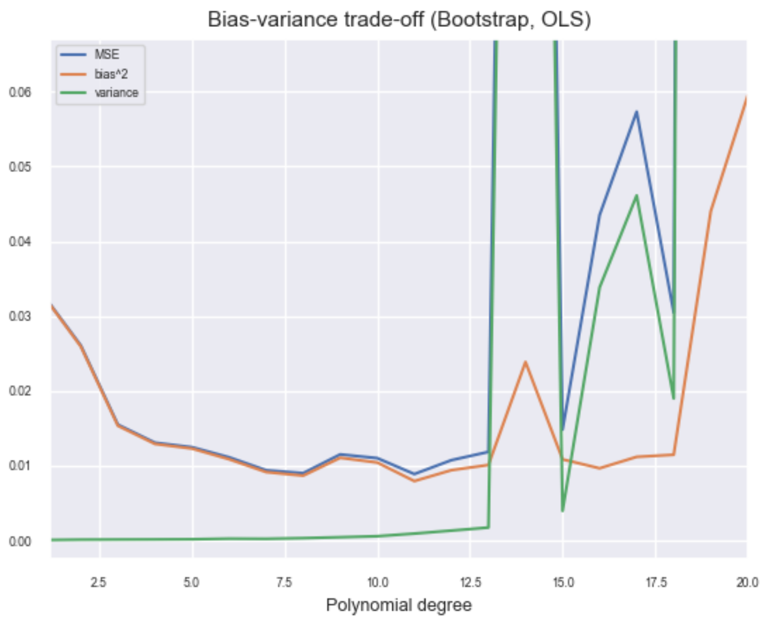


Figure 9: Bias-variance analysis for OLS regression, calculated using bootstrapping. This leads to a higher estimate for variance than the same calculation performed with cross validation. For the particular run shown, the estimates for variance, and even bias, are unstable for high order polynomials.

both methods it feels safer to pick a model from the lower end of this range, based on bias and variance analysis. Also, less complex models are faster to train. Alternatively, one could pick a higher degree Ridge model (12-16) and turn up regularization to for example 0.001.

But on the pure MSE, there was no strong benefit observed from regularization. This would kick in if a smaller data set was used or if the added noise level was set higher.

Due to computational constraints I had to limit Lasso regression to fifth order polynomials. With regularization set low enough, Lasso performed as well as Ridge for the complexities tested. It would have been interesting to test Lasso also for higher order models, but in the analysis done there was nothing to indicate Lasso would start to outperform the other methods.

4.1.1. Terrain Data

OLS and Ridge models with polynomial degree 8-18 works reasonably well, giving an MSE of roughly 10 meters. As for Franke's function, there were no strong beneficial effects from regularization on MSE. Again, Lasso regression was only tested for polynomial degrees up to five. The best MSE achieved was roughly 20 meters. But this was similar to Ridge for lower order models and it is suspected that Lasso models of higher degree would also perform better. This should be tested.

References

[1] Wikipedia contributors, Biasvariance tradeoff — Wikipedia, the free encyclopedia,

Appendix A.

```
def OLS_analytical(X, y):
    try:
        beta = np.linalg.inv(X.T@X)@X.T@y
    except np.linalg.LinAlgError as err:
        beta = np.linalg.pinv(X)@y
    return beta
```

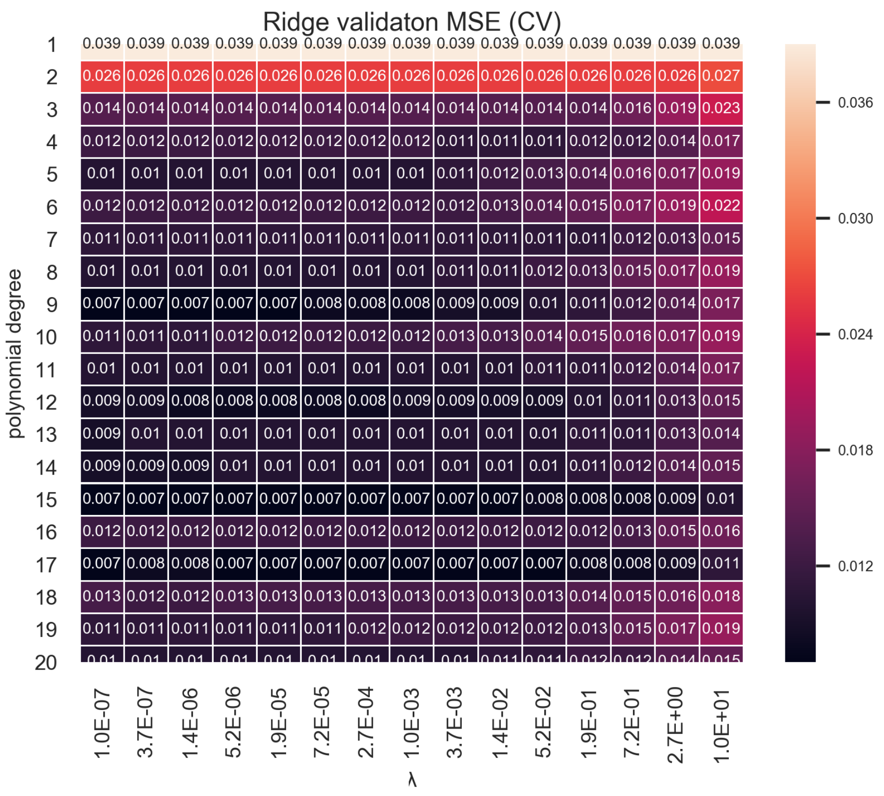


Figure 10: MSE on validation sets, using Ridge regression and cross validation. Higher order models with small λ perform better, but gains are limited after polynomial degree four.

Appendix B.

```
def cv(reg_func, X, y, k, X_test, y_test, _lambda=0):
    '''
    Performs k-fold cross validation on input design matrix x and target vector y.
    The estimator has to be given in variable 'reg_func'.
    Predictions are also calculated for the separate test set (X_test, y_test) in
    order to estimate bias and variance
    '''
    X_test_orig = X_test
    k_size = int(np.floor(len(X)/k))
    MSE_val = np.zeros(k)
    R2_val = np.zeros(k)
    y_predict_cv_val = np.zeros((k_size,k))
    y_predict_cv_test = np.zeros((len(X_test),k))
    y_cv_val = np.zeros((k_size,k))

    for i in range(k):
        #k-fold cross vaildation
        #splitting X into training and validation sets
        #no shuffling is done, assuming input is already shuffled.
        test_ind = np.zeros(len(X), dtype = bool)
        test_ind[i*k_size:(i+1)*k_size] = 1
        X_cv_train = X[~test_ind]
        X_cv_val = X[test_ind]
        y_cv_train = y[~test_ind]
        y_cv_val[:,i] = y[test_ind]
        X_cv_train, mu, stdev = standardize(X_cv_train)
```

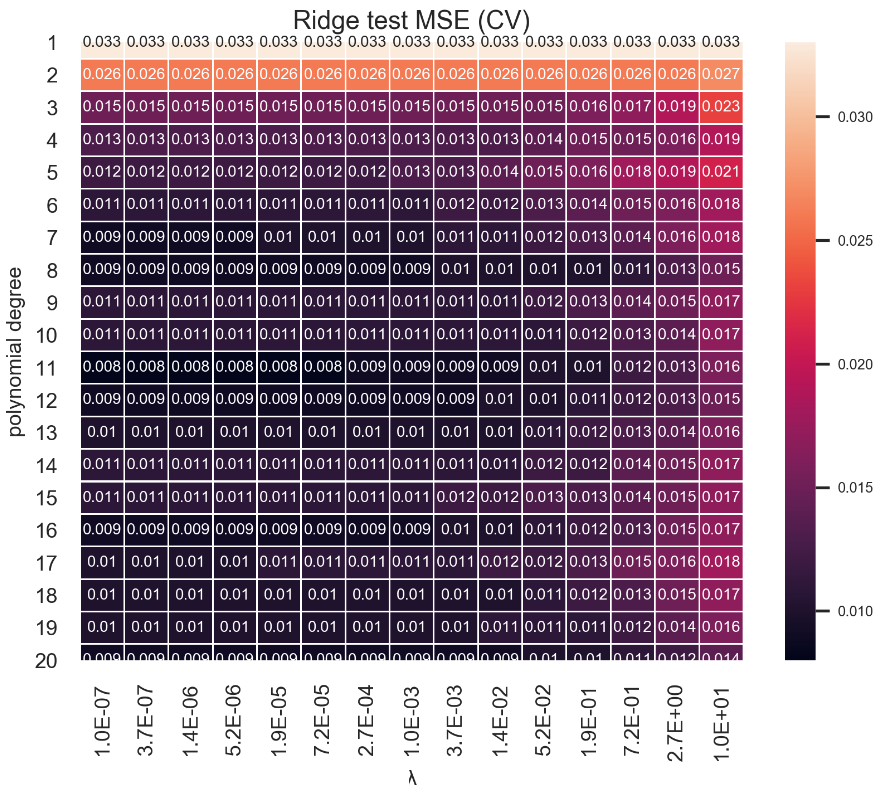


Figure 11: MSE on the separate 20% test set, using Ridge regression and cross validation. Again, higher order models with small perform better, but gains are limited after polynomial degree six.

```
        X_cv_val = (X_cv_val - mu)/stdev
        X_test = (X_test_orig - mu)/stdev
        if len(inspect.getargspec(reg_func).args)==3:
            beta = reg_func(X_cv_train, y_cv_train, _lambda)
        else:
            beta = reg_func(X_cv_train, y_cv_train)
        y_predict_cv_val[:,i] = X_cv_val@beta
        y_predict_cv_test[:,i] = X_test@beta
        #calculating MSE and R2 on the validation sets
        R2_val = r2_score(y_cv_val[:,i], y_predict_cv_val[:,i])
        MSE_val = mean_squared_error(y_cv_val[:,i], y_predict_cv_val[:,i])

    MSE_val = np.mean(MSE_val)
    R2_val = np.mean(R2_val)

    #calculating MSE, variance and (bias + random noise) on the separate test set
    MSE_test = np.mean(np.mean((y_test[:,None] - y_predict_cv_test)**2, axis=0, keepdims=True))
    variance_test = np.mean(np.var(y_predict_cv_test, axis=1))
    bias_test_plus_noise = np.mean((y_test[:,None] - np.mean(y_predict_cv_test, axis=1, keepdims=True))

    return MSE_val, MSE_test, R2_val, bias_test_plus_noise, variance_test
```

Appendix C.

```
#Code run testing polynomials up to degree 10 on Franke's function
bingo@bingo-G56JK:~/python/uio_ml$ python3 project1_Franke.py
Polynomial degree: 1
Polynomial degree: 2
```

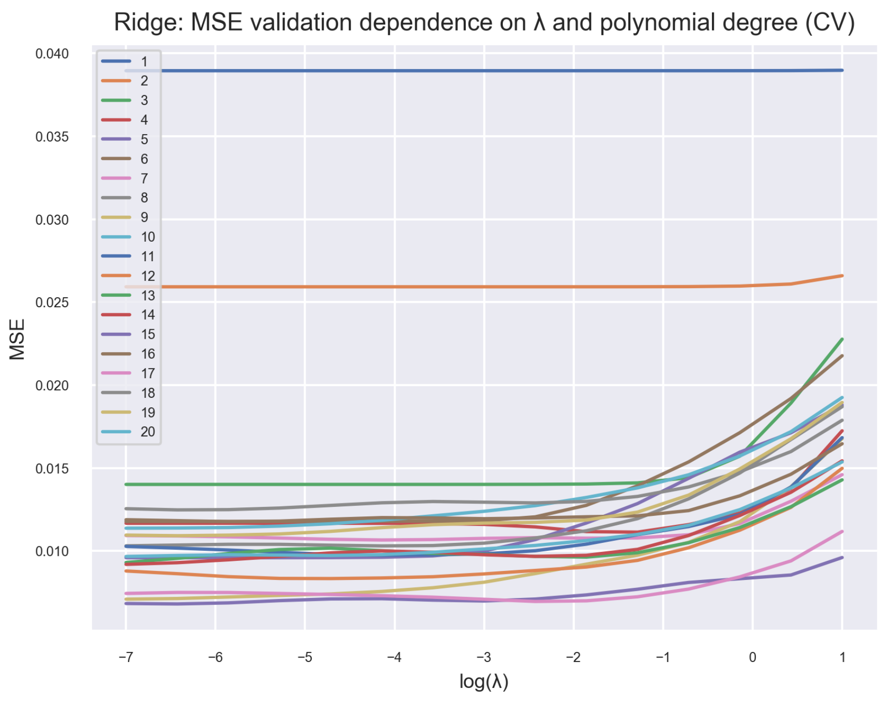


Figure 12: MSE on validation sets, using Ridge regression and cross validation. There is no strong benefit from regularization, although some models has a dip in MSE for certain values of λ.

```
Polynomial degree: 3
Polynomial degree: 4
Polynomial degree: 5
Polynomial degree: 6
Polynomial degree: 7
Polynomial degree: 8
Polynomial degree: 9
Polynomial degree: 10
Ridge best parameters: Degree=7, Lambda=1e-07
Ridge best MSE on validation set: 0.0101695, corresponding MSE on test set:0.0105639
```

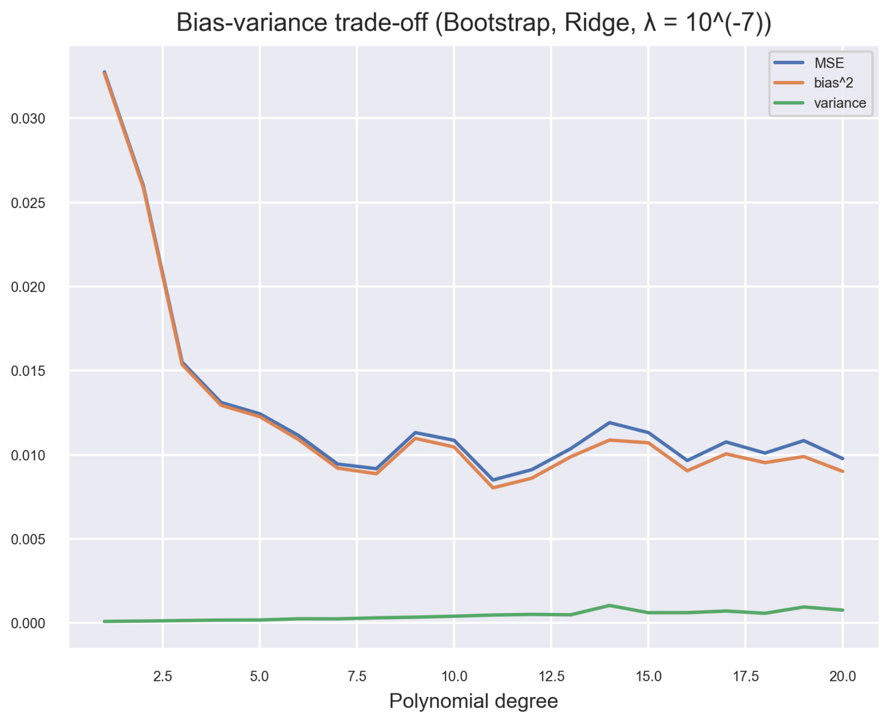


Figure 13: Bias-variance analysis for Ridge regression with $\lambda = 10^{-7}$, calculated using bootstrapping.

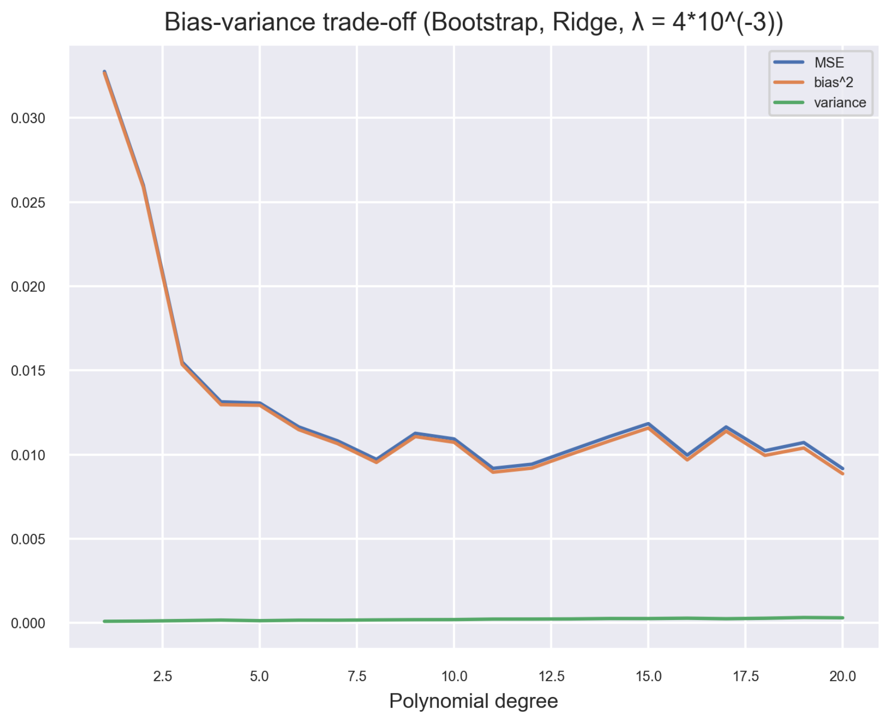


Figure 14: Bias-variance analysis for Ridge regression with $\lambda = 4 * 10^{-3}$, calculated using bootstrapping. Compared to figure 13 a more aggressive regularization is seen to minimize variance and increase bias for high order models.

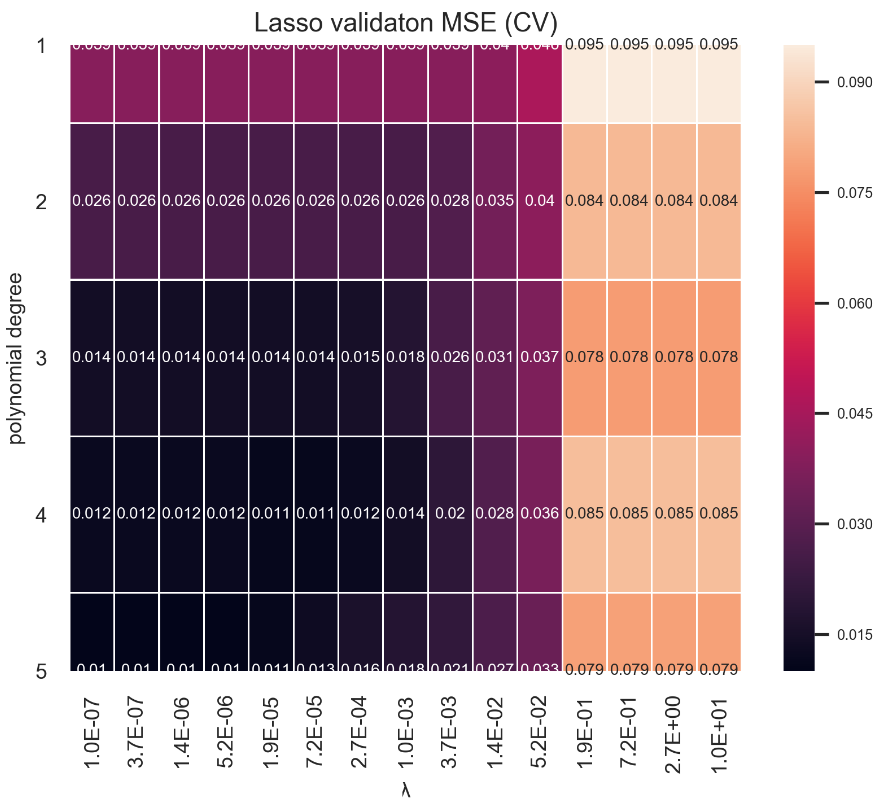


Figure 15: MSE for Lasso regression on validation sets using cross validation. Of the tested models, polynomial degree five performs best. There is no clear benefit from regularization. Regularization $\lambda > 0.1$ is clearly to aggressive.

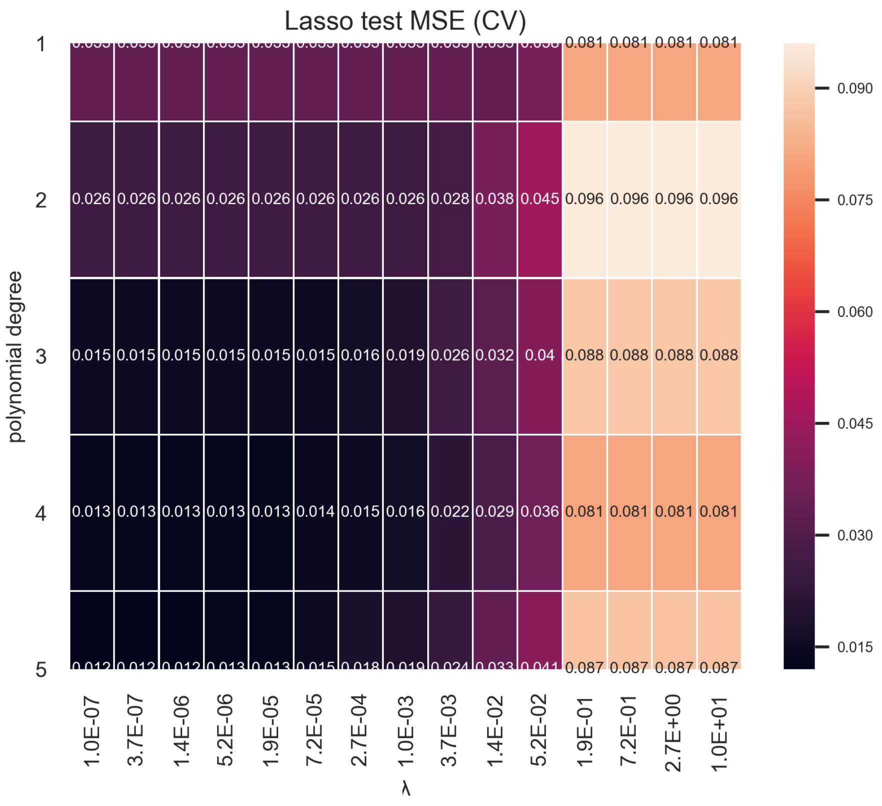


Figure 16: MSE for Lasso regression on the separate test set using cross validation. Confirming the results on the validation sets, polynomial degree five performs best. There is no clear benefit from regularization. Regularization $\lambda > 0.1$ is clearly to aggressive.

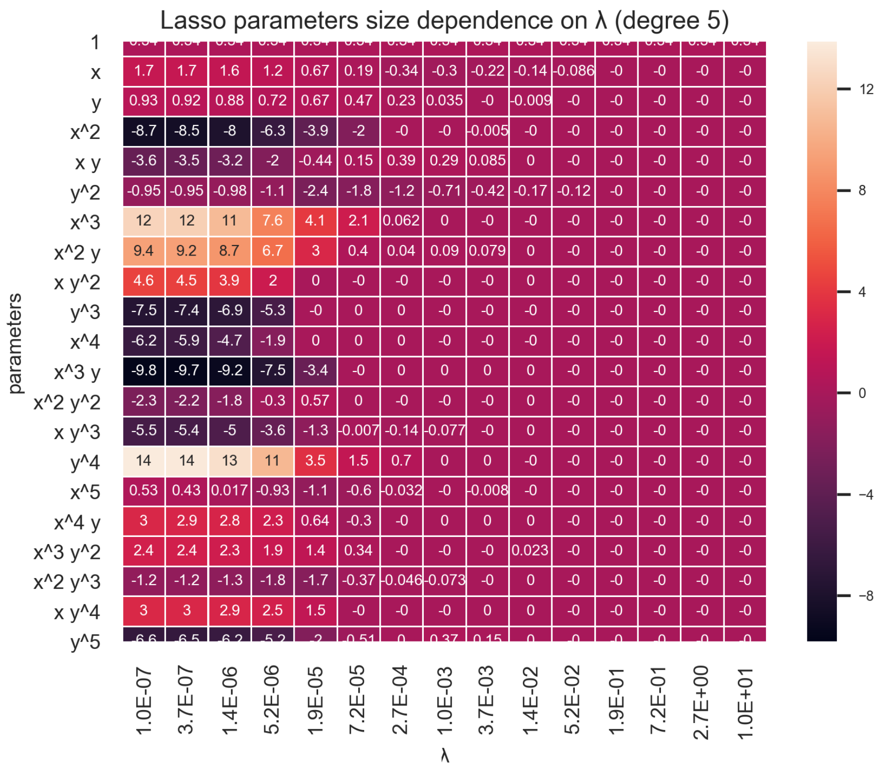


Figure 17: Lasso coefficients β for a fifth order polynomial model with varying regularization. Aggressive regularization $\lambda > 0.2$ leaves only the intercept nonzero.

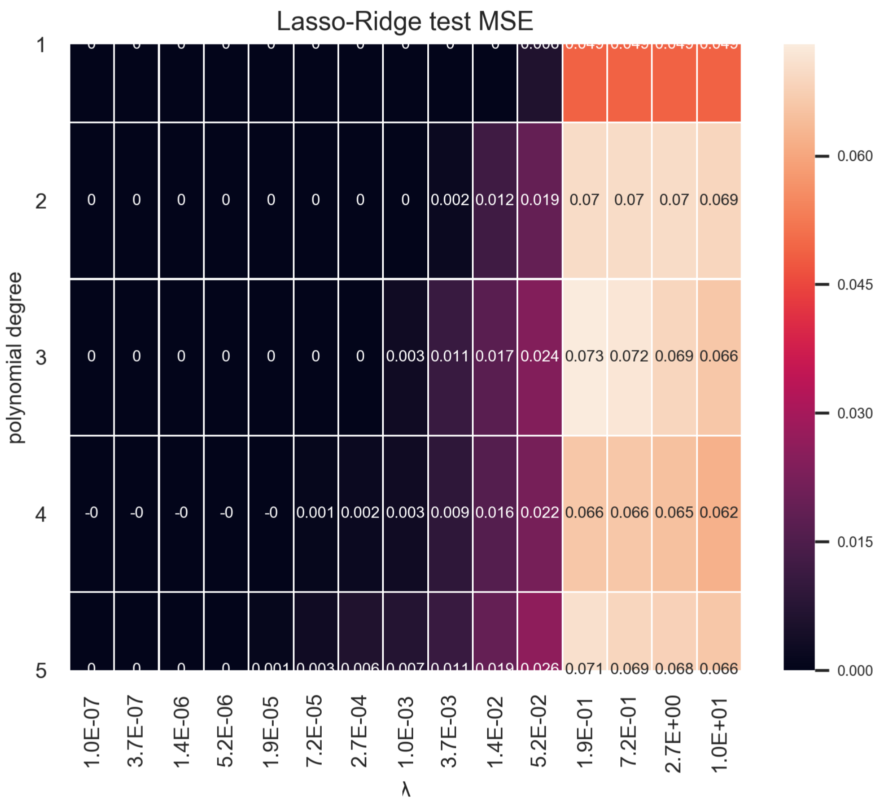


Figure 18: Difference between MSE for Lasso and Ridge regression. It shows almost identical results for smaller λ. Regularization $\lambda > 0.1$ is clearly to aggressive for Lasso.

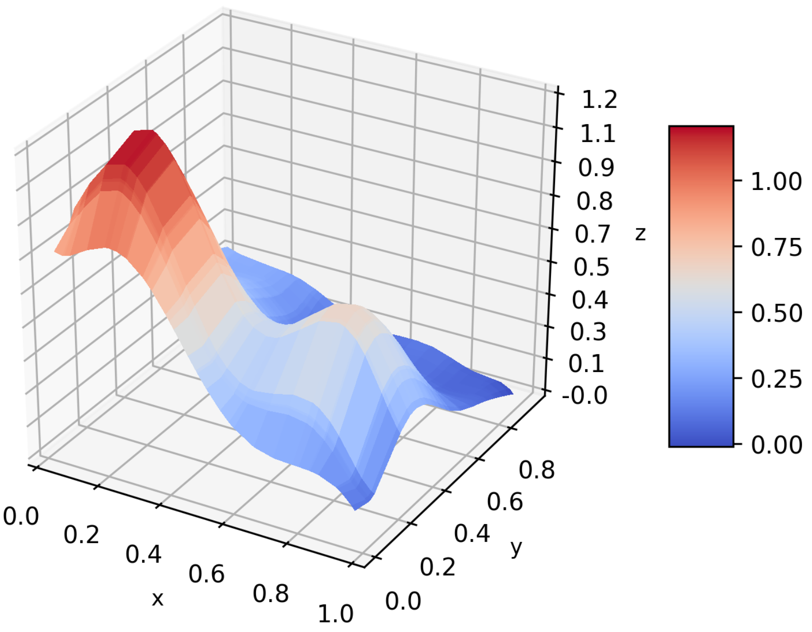


Figure 19: Franke's function estimate using the Ridge model that had lowest MSE on the validation sets.

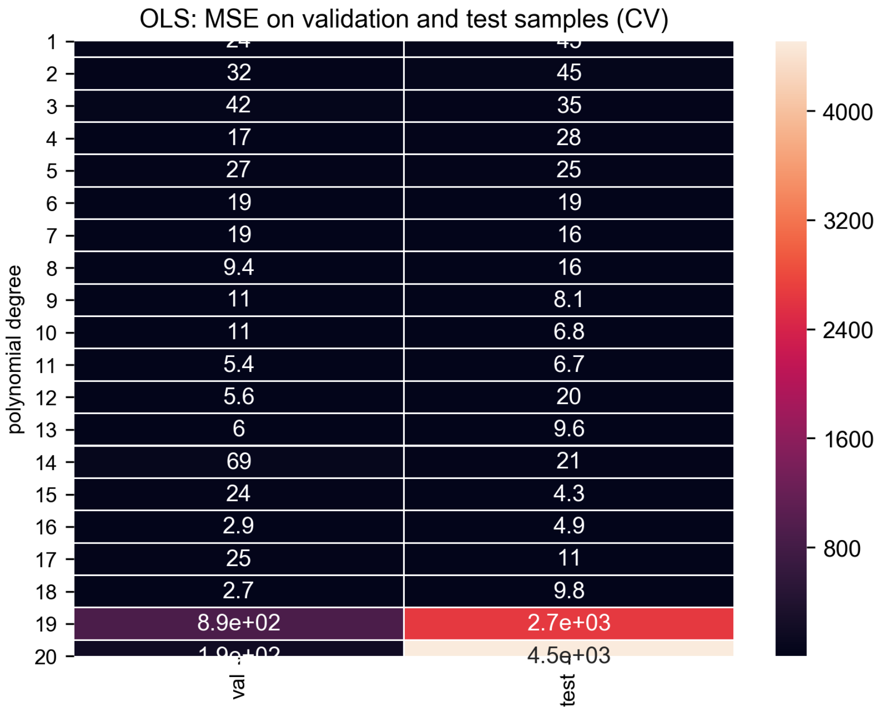


Figure 20: Average MSE calculated on validation sets and the test set using cross validation. Results are unstable due to relatively few data points, especially for higher order models. But overall models of order 9-18 performs well.

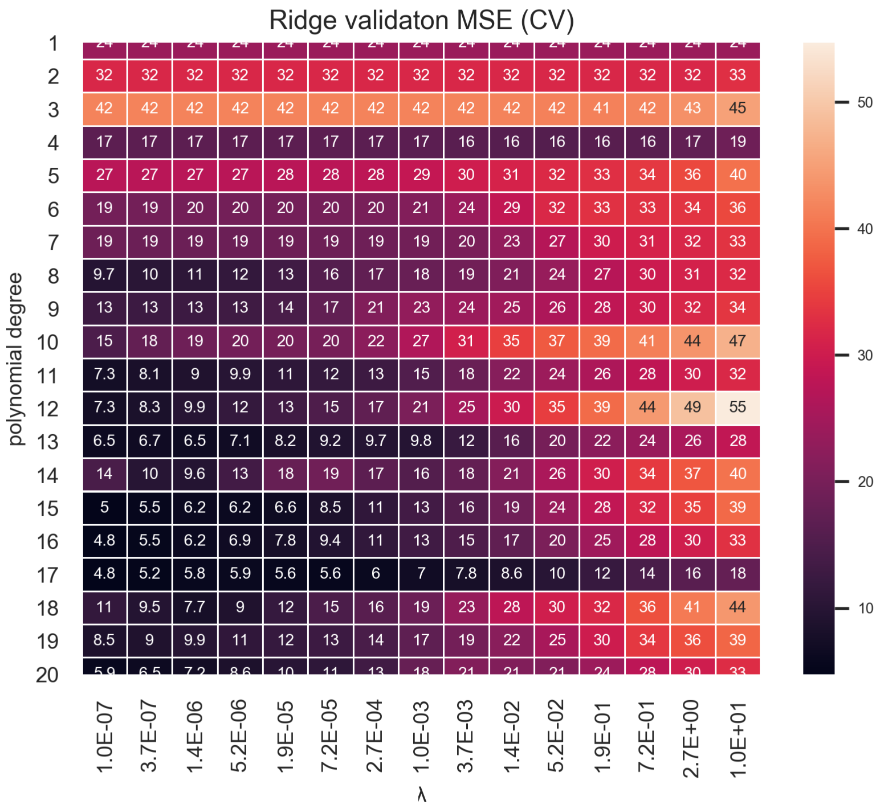


Figure 21: Average MSE for Ridge regression calculated on validation sets using cross validation. High polynomial order and low λ gives the best results.

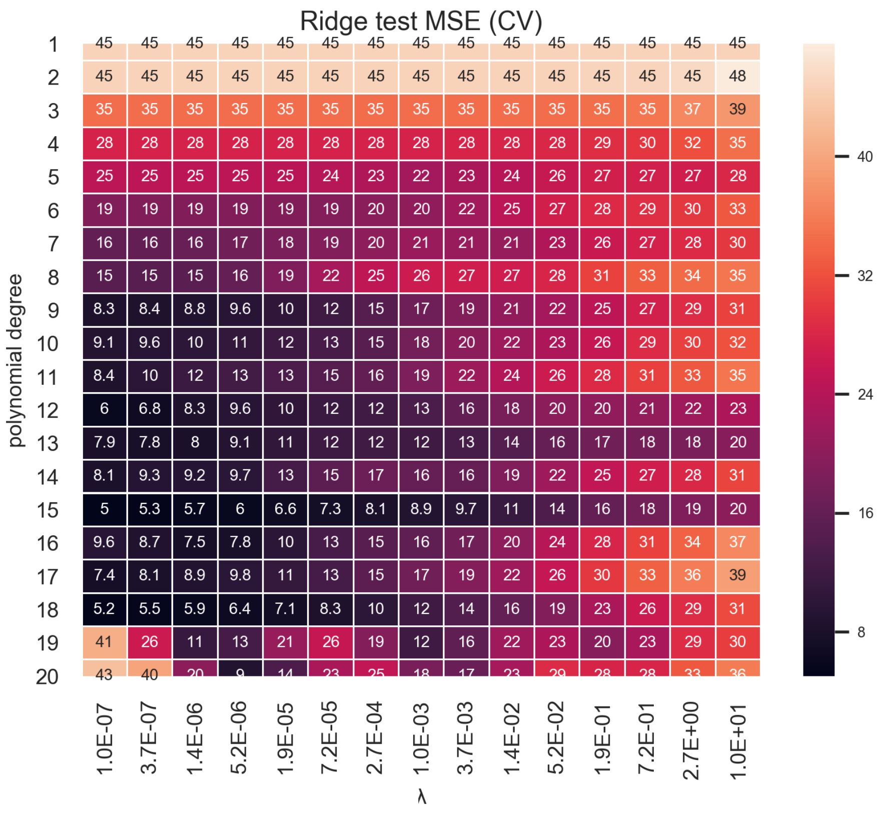


Figure 22: Average MSE for Ridge regression calculated on the separate test set using cross validation. High polynomial order and low λ gives best results. Here, but not on the validation set, worsening performance for polynomials 18 and 19 is seen. Model results are likely quite unstable at this high order as the number of explanatory variables approaches the number of data points.

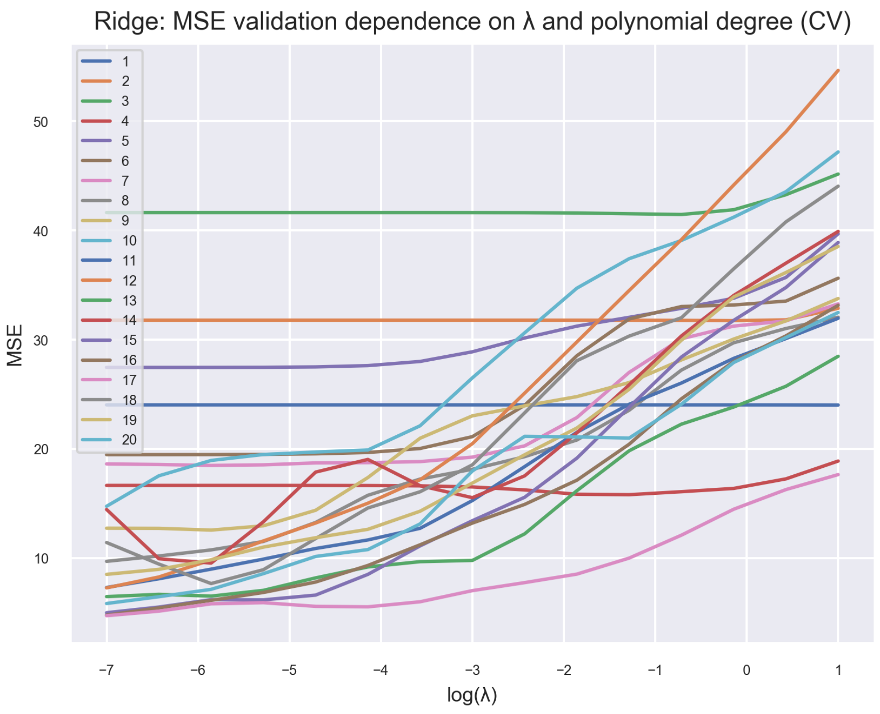


Figure 23: Average MSE for Ridge regression calculated on validation sets using cross validation. High polynomial order and low λ gives best results. There is no clear evidence of benefit from regularization.

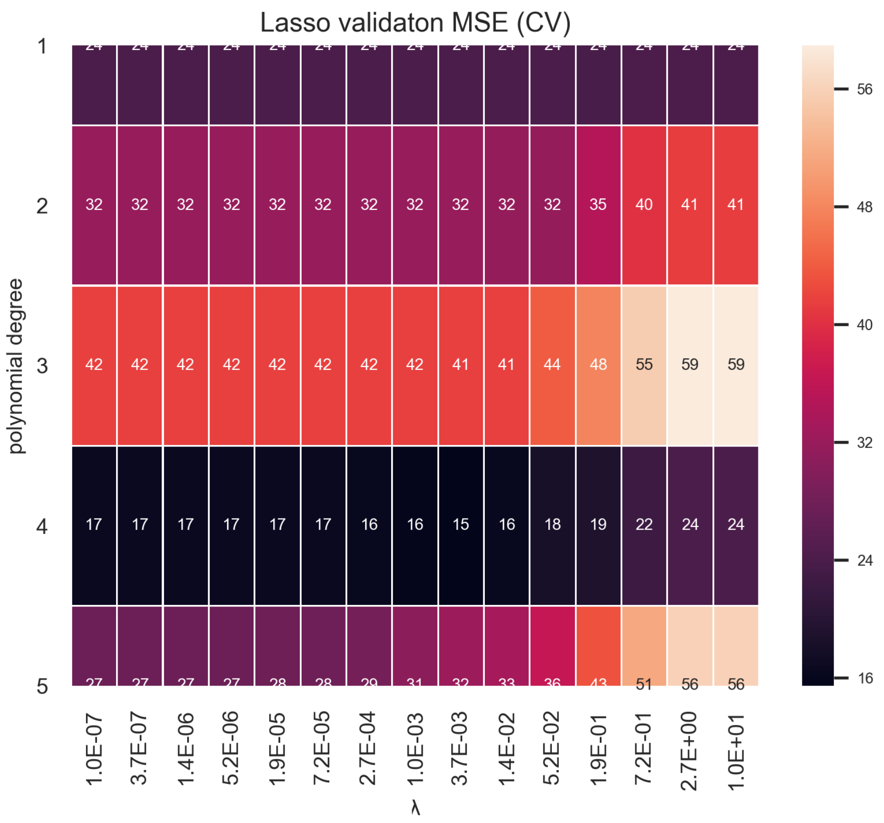


Figure 24: Cross validated MSE on validation sets for Lasso regression as function of polynomial degree and regularization. Of the tested models, polynomial degree 4 performs best on the validation sets. MSE rises sharply with regularization $\lambda > 0.2$.

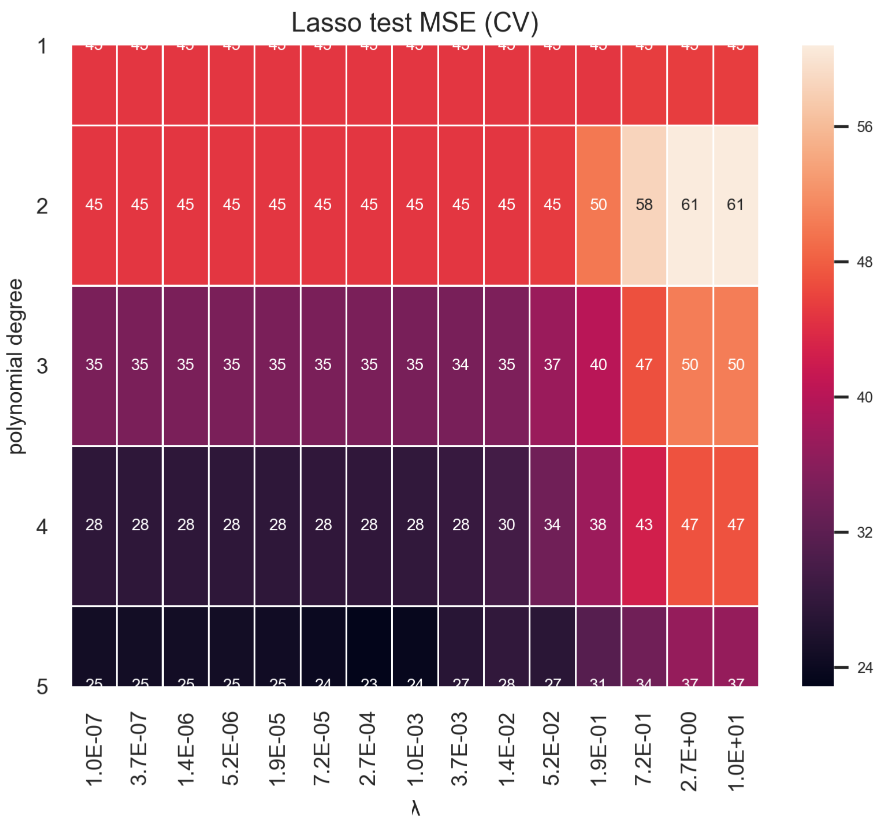


Figure 25: Cross validated MSE on the separate test set for Lasso regression as function of polynomial degree and regularization. Of the tested models, polynomial degree 5 performs best on the test set. MSE rises sharply with regularization $\lambda > 0.2$.

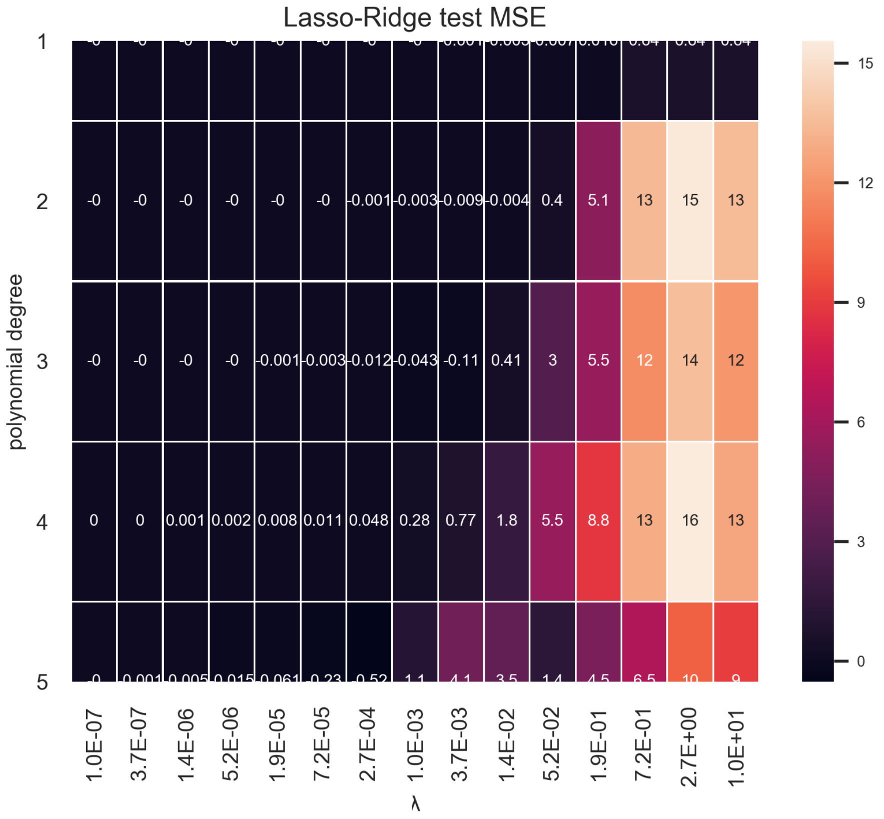


Figure 26: Difference between MSE for Lasso and Ridge regression. It shows almost identical results for smaller λ. Regularization λ > 0.2 is clearly to aggressive for Lasso.

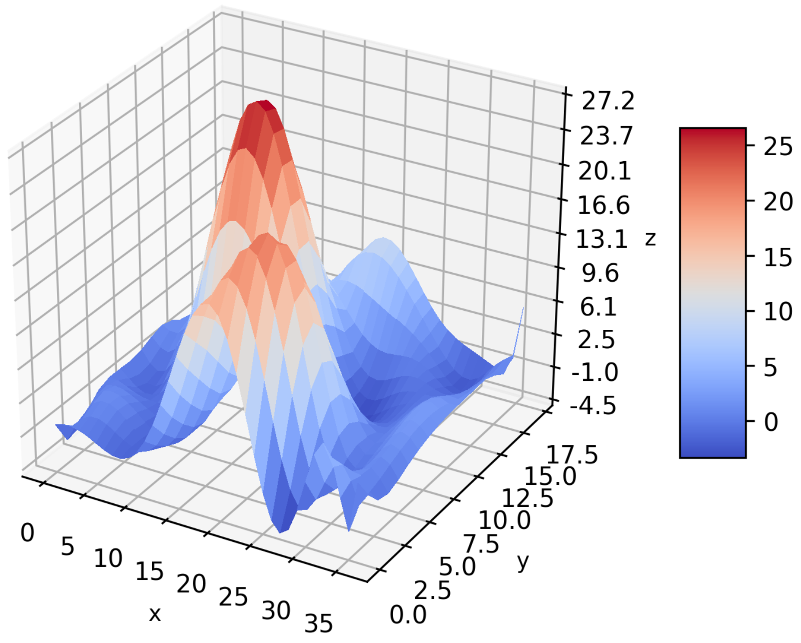


Figure 27: Terrain predictions using the Ridge model that had lowest MSE on the validation sets.