University of Oslo

Project 1

FYS-STK4155

José Luis Barreiro Tomé

Romain Corseri

October 7, 2022

Abstract:

Linear regression is the simplest form of supervised machine learning and it is generally the first approach tested in many applications. In this report, we assess the performance of Ordinary-Least-Squares (OLS), Ridge and Lasso (least absolute shrinkage and selection operator) regression on a synthetic dataset of 100 datapoints generated by the Franke function and contaminated by gaussian noise. Then we use the same approach parameterizing digital terrain data over Telemark (Norway) from the Shuttle Radar Topography Mission (SRTM). Different parameters were tested to observe the behavior of the models, mainly model complexity as polynomial degree, and the regularization parameter λ. The performance of the models was gauged through the resampling techniques of bootstrapping and cross-validation. Ordinary-Least-Squares method outperformed Ridge and Lasso regression in both our synthetic dataset and the real terrain elevation dataset. For the Franke function, we find that a polynomial of degree 4 is the best compromise between data misfit and complexity. Our OLS regression analysis shows that a polynomial function of degree 3 would fit the terrain data best, despite smoothing-out abrupt topographic variations, like deep riverbeds and valleys. We conclude that, despite being one of the simplest regression analysis, OLS is a powerful method to fit the Franke function. In turn, a more sophisticated, perhaps non-linear, regression method would be needed to fit elevation data with abrupt variations and outliers.

1. Introduction

Linear regression is perhaps the simplest form of supervised machine learning algorithm. However, the method conveys important concepts of regression analysis which allows us to go beyond the model that describes the data best.

In this project, we investigate the performance of ordinary least square (OSL), Ridge and Lasso regression methods for two datasets. First, we use a synthetic dataset generated by the Franke function (Franke, 1979), which is a two-dimensional function. Using those synthetic data, we can vary noise levels, number of data points, regularization parameters and thereby explore the various regression methods in detail. In the second part, we use real topographic data from Norway and apply similar regression analysis.

In the last part of the report, we discuss which regression method and resampling techniques are most adapted to fit our synthetic and real datasets. We conclude the report by advocating for a bi-variate polynomial model that best describe our synthetic and real observations in the desired uncertainty.

2. Data and Methods

2.1. Theory

2.1.1. Regression and resampling techniques

In this work, we use three linear regression methods, namely ordinary least square (OLS), Ridge and Lasso. Those regression methods assume that there exists a convex function that can approximate our data and that the Hessian matrix $X^T.X$ is invertible. In all three cases, the solution of the regression problem – i.e. the optimal predictor β that minimizes the cost function - can be expressed analytically:

- **OLS:** $\beta = (X^T.X)^{-1}.X^T.y$

  where X is the design matrix and y, the data

- **Ridge:** $\beta = (X^T.X + \lambda I)^{-1}.X^T.y$

  where λ is the hyper parameter and I, the identity matrix

- **Lasso**: $X^T.X\,\beta + \lambda\, sgn(\beta) = 2\,X^T.y$

  In the Lasso case, the solutions of the equation are found with an iterative solver, like the gradient descent. sgn() is a discontinuous function so the equation does not lead to a nice analytical solution for β.

The **bootstrap and cross-validation** are two re-sampling techniques underlain by the fundamental central limit theorem. The theorem states that the probability-density function (PDF) P(x) of the average of m random values with a corresponding PDF p(x) is a normal distribution whose variance is the variance of p(x) divided by m, the number of experiments.

In the bootstrap method, the experiment is the random drawing of numbers from p(x) with replacement. For cross-validation (CV), we split the dataset into k groups (so-called k fold number), being one of them the test data where the model is tested, and the rest the training dataset where the model is fitted. In a loop, every time a different test and train groups are set, and the averaged model error is returned.

2.1.2. Model assessment: Mean squared error and R2-score

Here, we use two metrics to assess the regression model ($\tilde{y}$ is the model prediction).

Mean square error: $MSE = \frac{1}{n}\sum_{i}^{n-1}(\, y_i - \tilde{y}_i\,)^2$

R2-score: $R2 = \; 1 - \frac{(\sum_{i}^{n-1}(\, y_i - \tilde{y}_i\,)^2)}{\sum_{i}^{n-1}(\, y_i - <\tilde{y}_i>)^2}$

2.1.3. Data scaling

In some instances, it may be important to re-scale the data for numerical stability and a fair penalization of the parameters. We scale the data with the StandardScaler() function of scikit-learn (Pedregosa *et al.* 2011), which works as follows:

$$y_{scaled} = (y - mean(y))/std(y)$$

2.1.4. Expectation values and variances

In linear regression techniques, we assume that there exists a continuous and convex function $\boldsymbol{f}$ and normally distributed noise $\boldsymbol{\varepsilon}$ that can perfectly describe our observations:

$$\boldsymbol{y} = f(\boldsymbol{x}) + \; \boldsymbol{\varepsilon} \qquad \boldsymbol{with} \qquad \boldsymbol{\varepsilon} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma^2})$$

f is a deterministic function and can be approximated by $\tilde{\boldsymbol{y}}$**:**

$$\tilde{\boldsymbol{y}} = \boldsymbol{X\beta} \;\; \text{where } \boldsymbol{X} \text{ is the design matrix}$$

Then we can express the expectation of $\boldsymbol{y}$**:**

$$\boldsymbol{E}[\boldsymbol{y}] = \boldsymbol{E}[\boldsymbol{X\beta}] + \boldsymbol{E}[\boldsymbol{\varepsilon}] = \boldsymbol{X\beta}$$

The variance of $\boldsymbol{y}$ can be expressed as follows:

$$\boldsymbol{var}(\boldsymbol{y}) = \boldsymbol{E}[\boldsymbol{y^2}] - \boldsymbol{E}[\boldsymbol{y}]^{\boldsymbol{2}}$$
$$\boldsymbol{var}(\boldsymbol{y}) = \boldsymbol{E}[(\boldsymbol{X\beta})^{\boldsymbol{2}} + \boldsymbol{\varepsilon}^{\,\boldsymbol{2}} + \boldsymbol{2\varepsilon X\beta}\,] - (\boldsymbol{X\beta})^{\boldsymbol{2}}$$
$$\boldsymbol{var}(\boldsymbol{y}) = (\boldsymbol{X\beta})^{\boldsymbol{2}} + \; \boldsymbol{2E}[\boldsymbol{\varepsilon}]\boldsymbol{X\beta} + \boldsymbol{E}[\boldsymbol{\varepsilon^2}] \; - (\boldsymbol{X\beta})^{\boldsymbol{2}}$$
$$\boldsymbol{var}(\boldsymbol{y}) = \boldsymbol{E}[\boldsymbol{\varepsilon^2}] = \boldsymbol{var}\;(\boldsymbol{\varepsilon}) = \boldsymbol{\sigma^2}$$

If the Hessian matrix $\boldsymbol{X^T.X}$ is not a singular matrix, then the expectation and variance of the optimal predictor $\boldsymbol{\beta}$ can be written**:**

$$\boldsymbol{E[\beta] = (X^T.X)^{-1}.X^{\,T}E[\tilde{y}]}$$
$$\boldsymbol{E[\beta] = (X^T.X)^{-1}.X^{\,T}.X.\beta = \beta}$$

$$\boldsymbol{var(\beta) = E[(\beta - E[\beta]).(\beta - E[\beta])^T]}$$
$$\boldsymbol{var(\beta) = E[((X^T.X)^{-1}.X^{\,T}.X.\,\tilde{y} - \beta).((X^T.X)^{-1}.X^{\,T}.X.\,\tilde{y} - \beta)^T]}$$
$$\boldsymbol{var(\beta) = (X^T.X)^{-1}.X^{\,T}.E[\,y.y^T].X.(X^T.X)^{-1} - \beta.\beta^T}$$
$$\boldsymbol{var(\beta) = (X^T.X)^{-1}.X^{\,T}.(X.\beta.\beta^T.X^T + \sigma^2).X.(X^T.X)^{-1} - \beta.\beta^T}$$
$$\boldsymbol{var(\beta) = \beta.\beta^T + \sigma^2\,(X^T.X)^{-1} - \beta.\beta^T = \sigma^2\,(X^T.X)^{-1}}$$

2.1.5. Bias-Variance trade-off

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

We can rewrite the above equation as:

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

Since f is a deterministic function:

$$\boldsymbol{E[y] = E[f + \varepsilon] = y}$$

Then we find the decomposition of the mean squared error:

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

2.2. Python programming and benchmark tests

In this project, we use the python programming language. To ensure reproducibility and repeatability of the results, we gathered all source codes in a GitHub repository (See appendix A.1 for program list and brief description). We make use of three essential open-source python libraires Matplotlib (Hunter 2007), NumPy (Harris *et al.* 2020) and Scikit-learn (Pedregosa *et al.* 2011). However, we do not use scikit-learn built-in functions to perform ordinary least square (OLS) and Ridge regressions but rather use them as benchmark to validate our own implementation and verify that they produce reasonable results (*LinReg()* and *RidgeReg*() in *Functions.py*). We use built-in scikit-learn function for the Lasso regression.

Note that the results presented in this project cannot be exactly reproduced as the *split_train_test()* or the addition of random noise to the data, will necessarily introduce a new input to each regression problem.

As a simple benchmark test for the OLS regression, we try to fit a model on a second order bi-variate polynomial like $f(x) = 1 + x + y + xy + x^2 + y^2$ and check if the regression results return $\beta_0 = \beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5 = 1$ for problem complexity of degree 2 or beyond.

2.3. Data

2.3.1. Franke Function

The Franke function is a function of two variables where $(x, y) \in [0,1]$ and is expressed as follows:

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

The Franke Function is depicted in Fig. 1 for two cases, with and without normally distributed noise. Note that when adding a normally distributed noise with a variance of 1, the noise dominates the data and cannot be used for regression purposes. We will therefore use a normally distributed noise with a variance of 0.1.

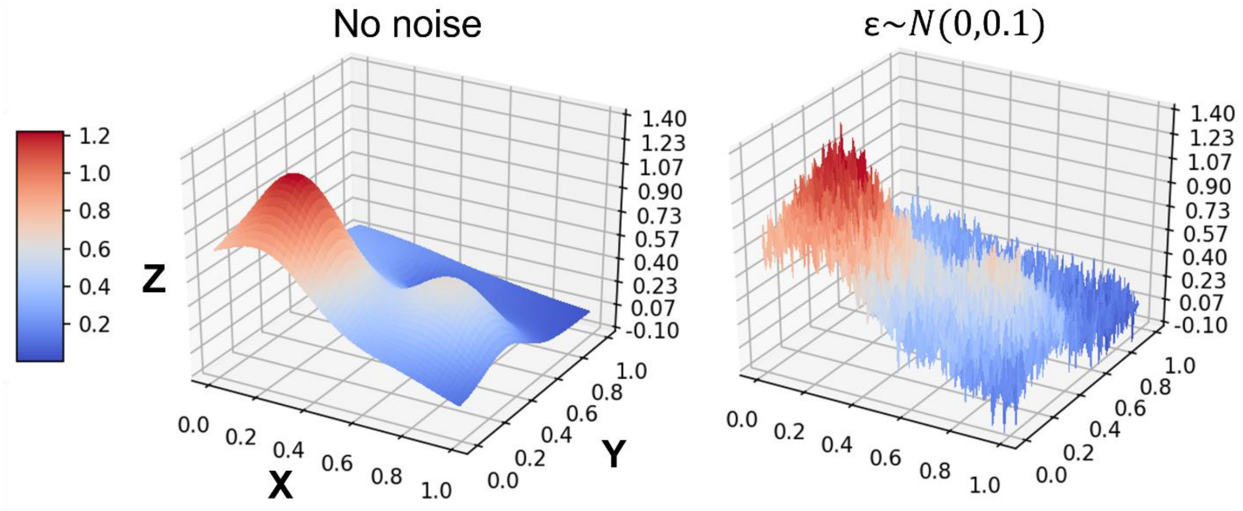


Fig. 1 3D visualization of the Franke Function without added noise (left panel) and with normally distributed noise with variance 0.1 and mean 0 (right panel).

2.3.2. Topographic data from Norway

After the code development of different regression and resample technics from previous steps, it becomes useful to look at real data. For this purpose, a digital elevation raster image is provided and used for fitting all different models. The dataset is located in Norway, on the Telemark region (Fig. 2) , and it was surveyed by the Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global (https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1).

The resolution of this USGS satellite mission is 1 arc-second (30 meters). The raster image is downsampled to 1000x1000 points by selecting the upper left corner, focusing on the Møsvatn Austfjell conservation area (Fig. 2).

A Standard Scaler is used to normalize the data. The regression analysis was initially performed up to a 10th order polynomial degree. However, the model errors escalated largely after the 5th order polynomial. Therefore, all the calculations were limited up to the 5th polynomial order (as tested in previous sections with the Franke function).

To evaluate which model fits the elevation data best, it was used a cross-validation resampling technique with 10 k-folds.

In Ridge and Lasso regression, we scan through 9 different values of the hyperparameter λ. This parameter is not learned by the model and is tested in each case to find the one that minimize the model error.

Several figures are produced to compare the mean square error for train and test dataset, depending on the complexity of the model (polynomial degree), and/or the hyperparameter λ. Finally, a 3D representation of the fitted data is produced and compared with the original dataset, for some selected polynomial degree and λ. All of this would help deciding which model performs better and under which parameters.

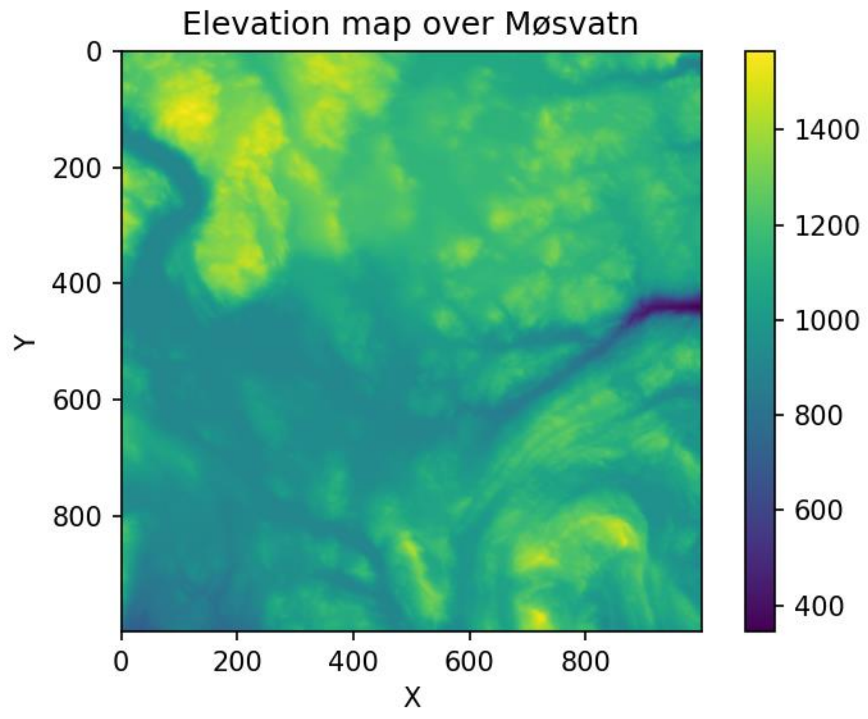


Fig. 2 Elevation map over Møsvatn (Data courtesy of USGS https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1).

3. Results

3.1. Regression results for the Franke function

3.1.1. OLS

We have implemented our own version of the OLS regression in a function called *LinReg()* in *Function.py*. We have compared the results of our own OLS implementation (*02_OLS_owncode.py*) with Scikit-learn (*OLS_scikit.py)* to fit a bi-variate polynomial function of degree 2, $f(x) = 1 + x + y + xy + x^2 + y^2$ using n=100 data points. The results of the benchmark tests are provided in Fig. A1 in appendix 5.2 and confirm the validity of our implementation of OLS for a simple regression problem. With our OLS implementation verified and validated, we move on to the regression of the Franke function.

On Fig. 3, we show the results of the OLS regression of the Franke function with an added normally distributed noise of variance 0.1, with 100 data points and problem complexity ranging from 1 to 5. We display two model metrics, the MSE and the R2-score. The data is split randomly into 80% training and 20% testing sets. We observe that the training MSE decreases monotonically towards 0 (or alternatively, R2 increases monotonically) with increasing model complexity whereas the testing MSE behaves more erratically. A minimal test MSE is obtained for a polynomial of degree 4. The high test MSE for degree 5 is an indication of overfitting.

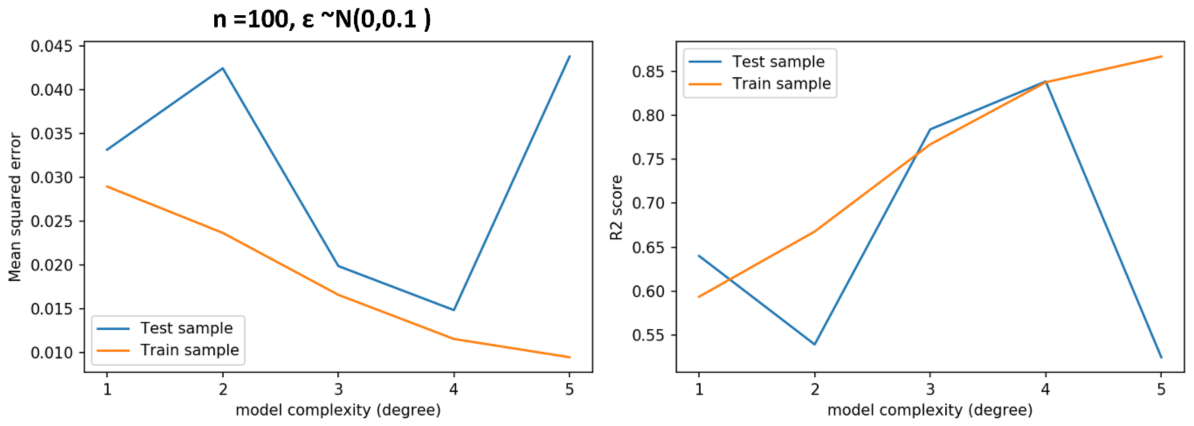


Fig. 3 Mean square error (left) and R2-score of the OLS regression model up to polynomial degree of 5

On Fig. 4, we plot the optimal $\hat{\beta}$ coefficients for problem complexity from degree 1 to 5. For better readability, we plot on Fig. 4b the optimal $\hat{\beta}$ coefficients with their respective uncertainty (standard deviation calculated with *Beta_std()* in *Functions.py*) only for degree 1, 3 and 5. We observe that the variances of $\hat{\beta}$ coefficients increase dramatically for degree 5, for β_3 and higher order coefficients.

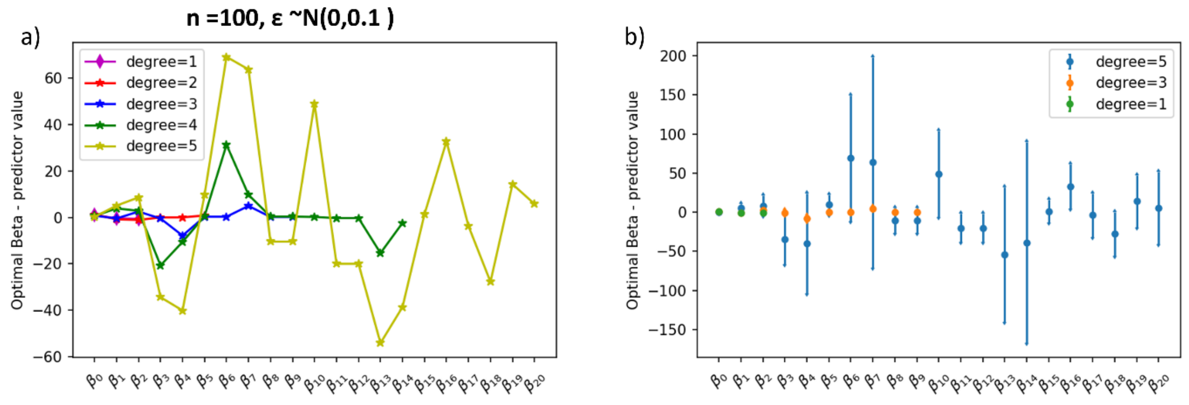


Fig. 4 Optimal B parameters of OLS regression models for polynomial degree 1 to 5 (left). Optimal B parameters and their related uncertainties for degree 1, 3 and 5.

We use *03_OLS_bootstrap.py* to produce bias-variance trade-off analysis for OLS. The decomposition of the MSE test error (see section 2.1.5) is useful as one can observe that the variance of the model increases with problem complexity (Fig. 5). It means that a small variation of the training sets could give a much different polynomial of degree 5. In Fig. 5, we observe that the number of data points has a strong impact on the result. Indeed, increasing the amount of data to n=1000 and bootstrap=750 will considerably decrease the variance of the model (following the central limit theorem) at higher complexity and the bias term becomes dominant in the error. For 100 datapoints, we see that a polynomial of degree 4 represent a good compromise between bias and variance of the model. For 1000 datapoints, a polynomial of degree 5 becomes acceptable.

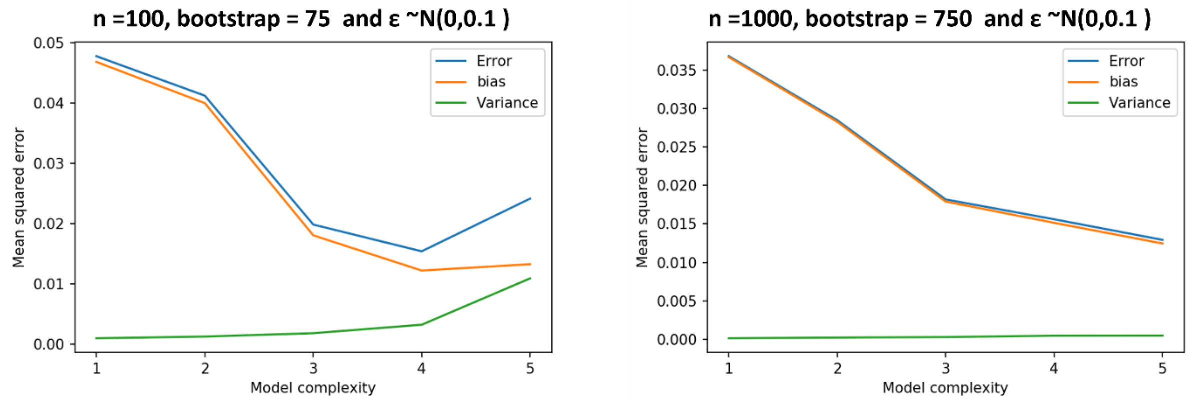


Fig. 5 Bias-variance trade-off of the OLS regression model with bootstrap resampling technique, for 100 datapoint and 75 bootstraps (left) and 1000 data points and 750 bootstraps.

We use *04_OLS_cv.py* to produce OLS errors with cross-validation (Fig. 6). We observe that the increasing number of folds from 5 to 10 does not impact the error achieved by the OLS regression. Overall, we see that the results from cross-validation and bootstrap resampling are very similar outcomes in terms of model error.

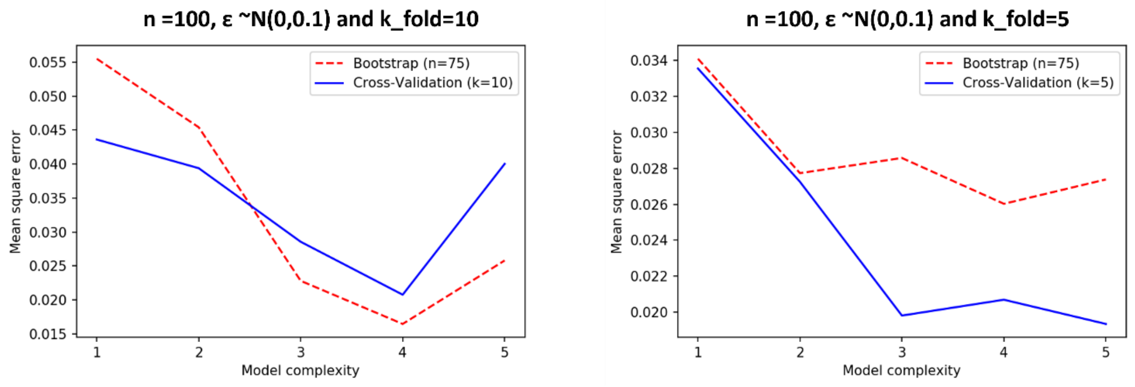


Fig. 6 Comparison of OLS regression results for 75 bootstraps and cross-validation with 10 folds (left) and 5 folds (right).

3.1.2. Ridge and Lasso

When performing Ridge and Lasso regression, we introduce a second parameter (the so-called hyperparameter λ) in the optimization problem. Therefore, we first investigate the effect of λ on the regression performance for a given problem complexity. In Fig. 7, we use *05_Ridge_Lasso_MSE_vs_lambda.py* for a polynomial of degree 4. When the noise level is low (variance=0.1) and λ gets closer to 0, the error decreases and converges to an irreducible value. The Lasso method gives a higher error than Ridge's for such noise level and the optimal λ parameter is also the closest to zero. In contrast, when the noise level in the data increases (variance=0.3), the behavior of the error changes fundamentally as an optimal λ parameter exists and minimizes the test error. For Lasso, λ=10^{-3} is optimal whereas for Ridge, we find that for λ=1 the test error reached a global minimum.

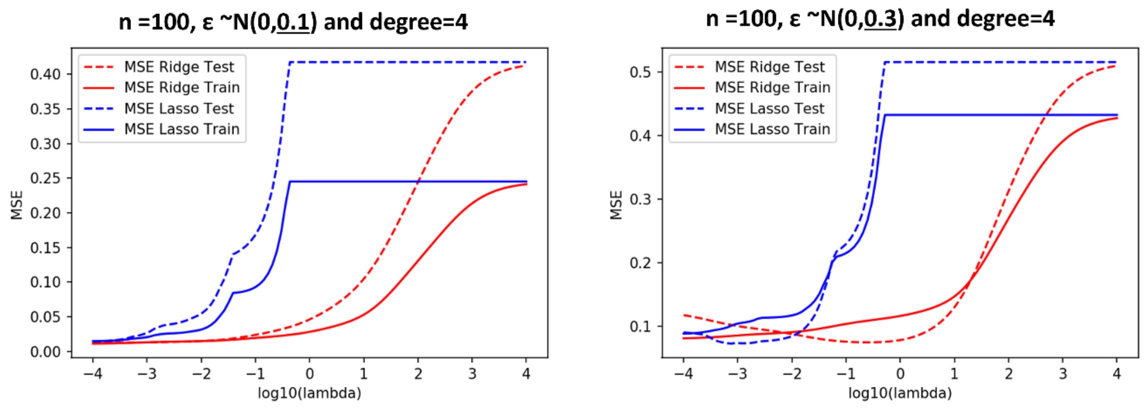


Fig. 7 Ridge and Lasso regression model error as function of hyperparameter Λ for a polynomial degree of 4 and two level of noise, variance=0.1 (left) and variance=0.3 (right).

A grid search on optimal λ and polynomial degree for minimizing the Ridge model error (with bootstrap) is implemented in *06_Ridge_bootstrap.py.* The results are presented in heatmaps in Figs. A2 in appendix 5.2 and here, we reproduce the bias-variance trade-off for three values of λ that gives a meaningful error reduction (Fig. 8). The same grid search procedure is applied to Lasso regression models (*08_Lasso_bootstrap.py*) with heatmap provided in Fig. A3 (appendix 5.2) and bias-variance trade-off analysis for a selection of λ (Fig. 8). A general observation is that the variance obtained for Ridge and Lasso models with bootstraps are lower than the one from OLS for the same number of data points and noise level (Fig. 5). However, the OLS error reaches smaller values, meaning that the Lasso and Ridge models have a higher bias than OLS models. Comparing Lasso and Ridge models (Figs. 8 and 9), we find that the Ridge regressions converges to lower error than Lasso's albeit a higher variance. The Lasso model errors look to be dominated by the bias in our cases.

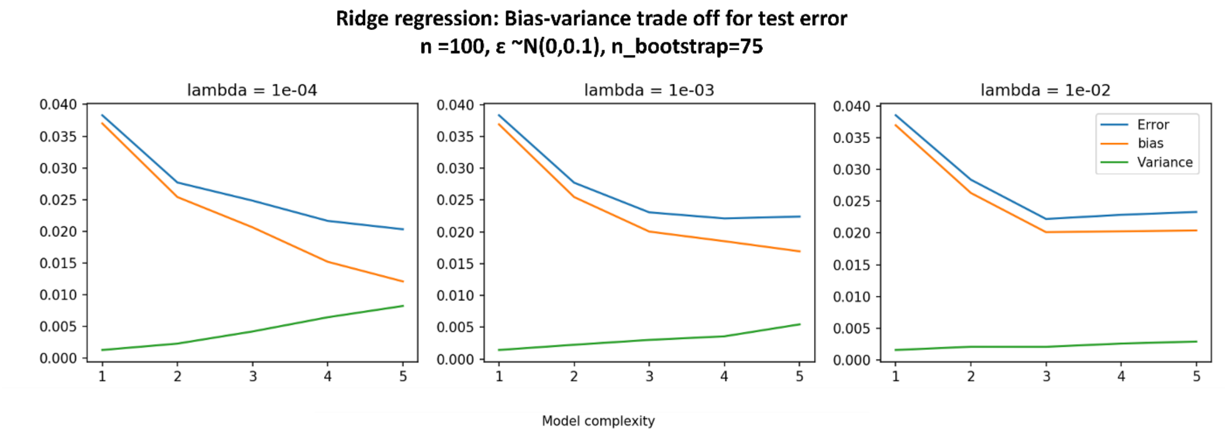


Fig. 8 Bias-variance trade-off for Ridge regression error with bootstrapping for three hyperparameters

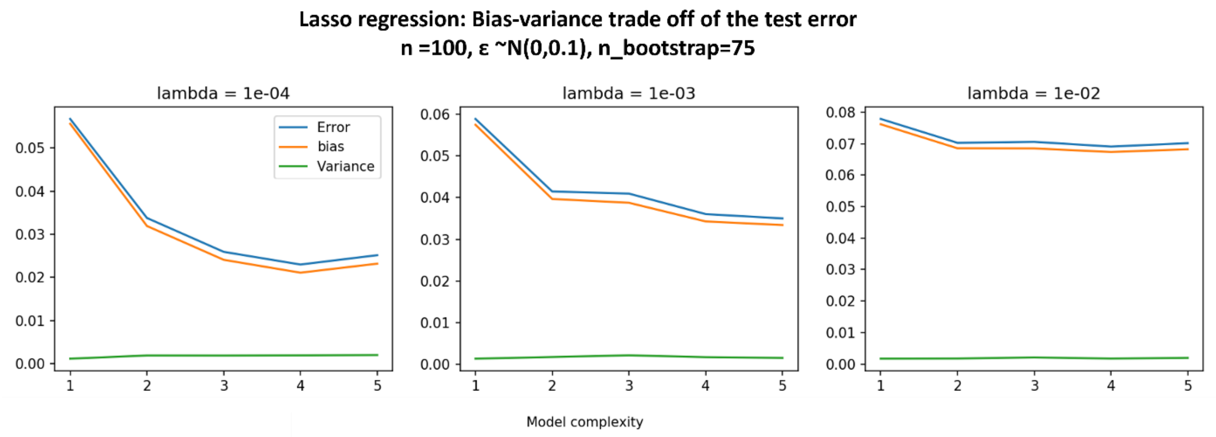


Fig. 9 Bias-variance trade-off for Lasso regression error with bootstrapping for three hyperparameters

We perform a grid search on optimal λ and polynomial degree for Lasso and Ridge methods but this time, using cross-validation resampling with 10 folds. The heatmaps of the grid search are provided in Figs. A2 and A3 for Ridge and Lasso methods, respectively. Again, we select three values of hyperparameters λ and investigate the development of the model error for each regression methods (Fig. 9). In both methods and in line with observations made in Fig. 7, the lowest model error are reached for the smallest hyperparameter $\lambda=10^{-4}$. However, we observe that Ridge regression models are outperforming Lasso's by systemically reaching lower MSE errors. These statements are only valid for a given noise level in the data (ε~N(0,0.1)).

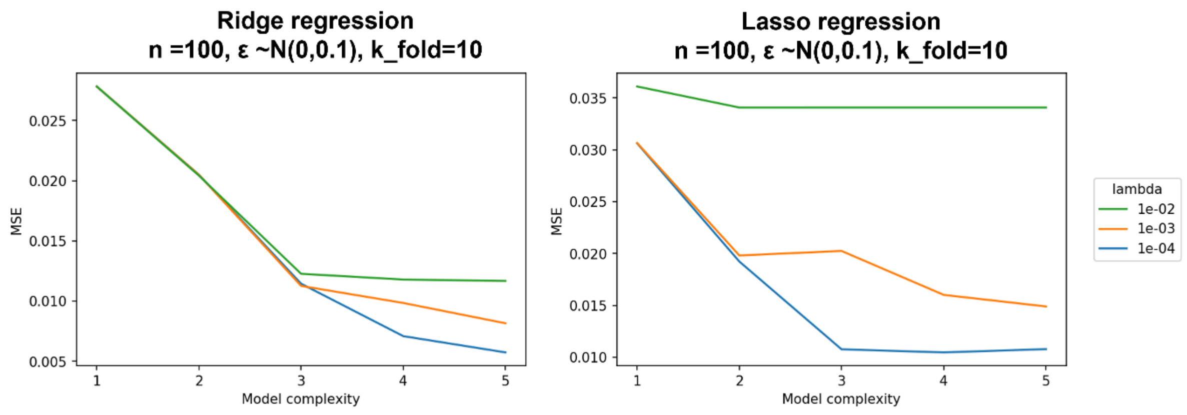


Fig. 9 Ridge (left) and Lasso (right) regression errors with cross-validation using k=10 folds for three hyperparameters.

3.2. Regression results for topographic data from Norway

In the script *10_Terrain.py*, Cross-validation method with Kfold=10 was performed for polynomial degrees 1 to 10 on a loop, and for a range of hyperparameter λ in the case of Ridge and Lasso. The behavior of the obtained mean squared error (MSE) versus polynomial degree on the OLS regression shows us that the MSE for the test datasets increases with increasing degree of complexity, especially with a degree higher than 5, i.e the model is overfitting Therefore, a smaller polynomial degree should be chosen. In other words, the fitted model would work better with higher complexity on the train dataset, but the same model applied to another dataset would give worse results. Polynomial degree 1 to 5 will be assessed in OLS. For Ridge and Lasso regression, this relation is different, and we can assess the degree of complexity from 1 to 10 (see Fig. 10).

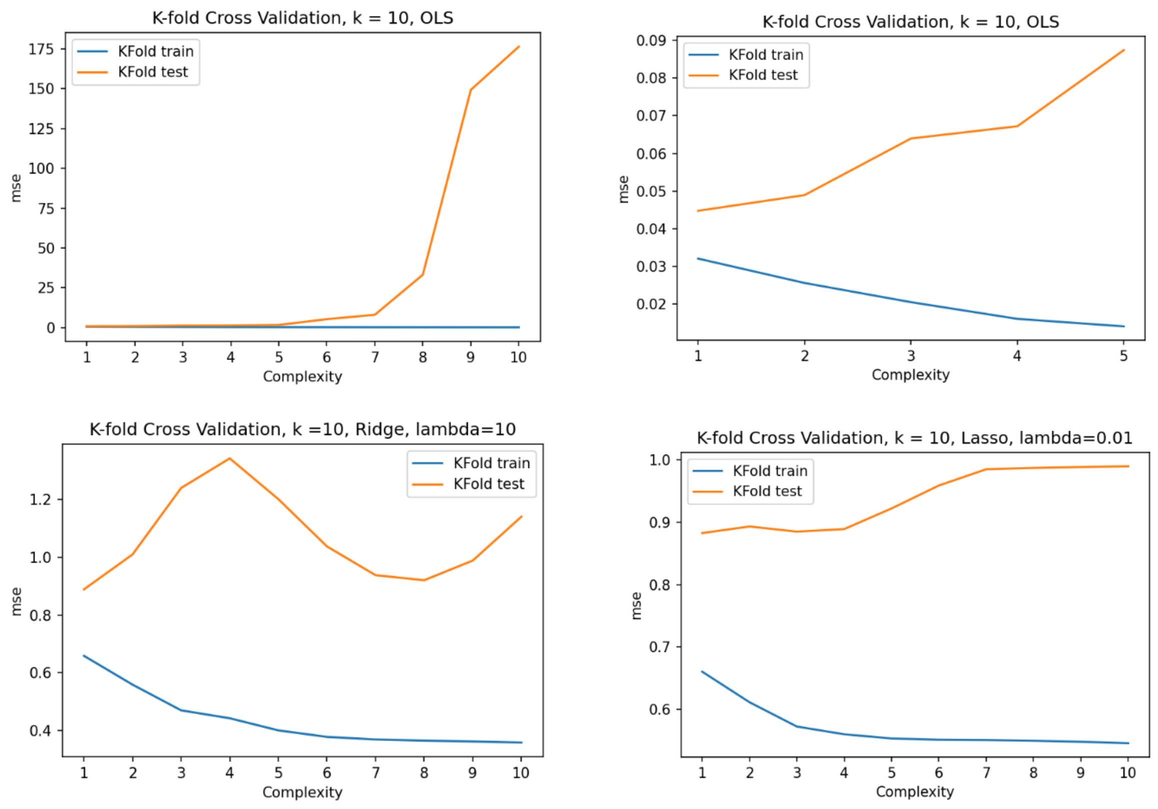


Fig. 10: OLS, Ridge and Lasso Cross-validation (10 folds) MSE for different polynomial degrees for train and test dataset.

When we apply the regression model to the whole dataset and visualize it in a 3D map, we can compare it with the original terrain dataset and to other regression methods.

For Ordinary-Least-Squares, a degree of complexity of 3 was chosen as a good compromise between the train/test MSE and the visualization of the data.

In the case of Ridge regression, and after some testing, the range for λ was set between 10^{-4} and 10^{4}, conducting 9 tests on the logscale (10^{-4}, 10^{-3} … 10^{4}). In this case a polynomial degree of 4 and a λ of 10^{2} was chosen.

The same approach as in Ridge was conducted for Lasso regression. After some testing, the chosen range of the hyperparameter λ for testing was 10^{-6} to 10^{2}, while the chosen optimal parameters were polynomial degree 4 and λ 10^{-1}.

Fig. 11 shows the performance of each λ regarding MSE when increasing the complexity of the model, both as a line and in a numerical way, colored in a heatmap for easy identification of low/high MSE values.

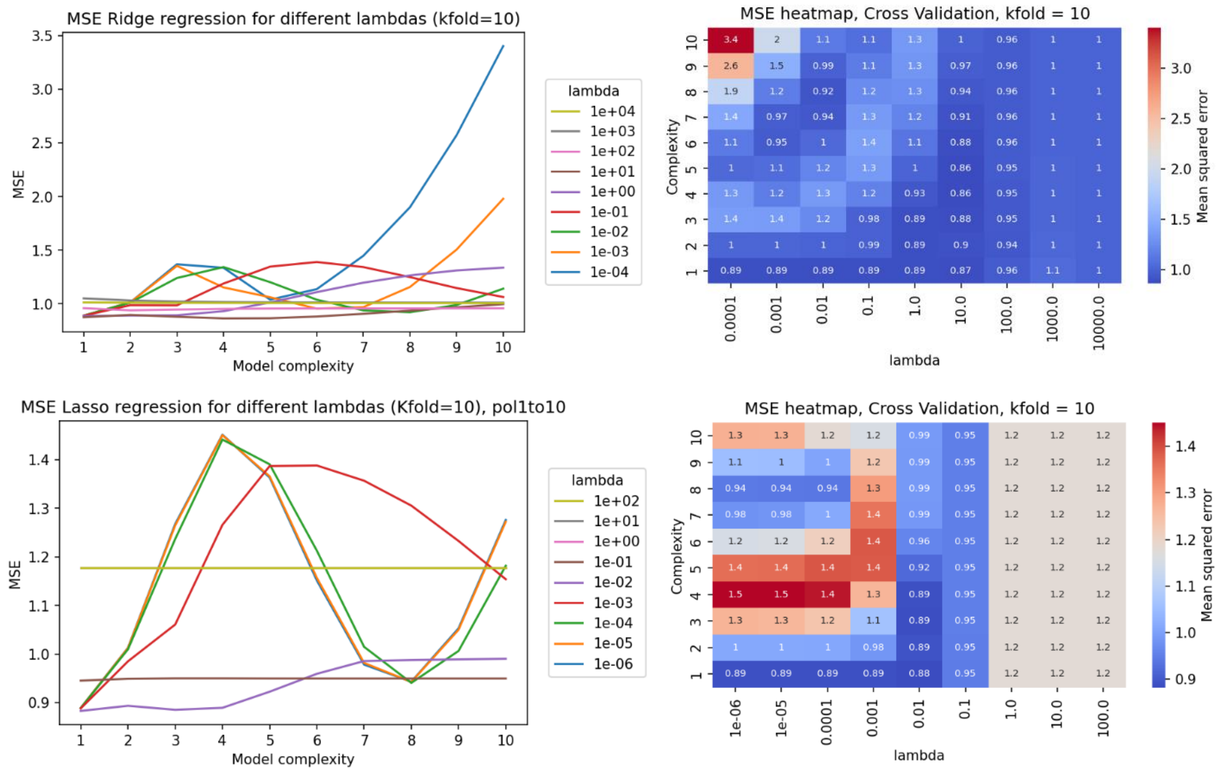


Fig. 11: Ridge (higher) and Lasso (lower) regression MSE of each tested λ for different polynomial degrees, showed as a line (left) and heatmap (right).

4. <u>Discussion and Conclusion</u>

4.1. What is the best regression method to fit the Franke Function?

We find that the best regression model for a dataset made of 100 data points generated from the Franke Function is a bi-variate polynomial of degree 4 whose coefficients are determined with the OLS method (Figs. 3, 4 and 5).

Overall, the Ridge and Lasso methods converge to a slightly higher test error than OLS and perform better as the hyperparameter gets closer to 0, thereby approaching the OLS solution. This conclusion no longer applies when the noise level has a variance of 0.3 (Fig. 7) and we perceive that Lasso and Ridge methods would be more competitive as we move to noisier data environments with outliers and exotic data points.

Finally, we chose the polynomial of degree 4 over a polynomial of degree 5 because of the very high variance and test error, clearly indicating an overfitting of the data. To our opinion, a polynomial degree of 3 would be underfitting as the training error does not reach an acceptable level (Fig. 3).

The cross-validation and bootstrap resampling techniques for OLS regression analysis have similar outcomes in terms of model metrics (Fig. 6). However, we would rather use the cross-validation technique because it is less costly computationally than bootstrapping for the error achieved given problem complexity from 1 to 5.

Finally, we claim that there is no need to rescale the data generated from the Franke Functions. The outcome of the OLS regression is similar with and without rescaling. This is because the regression problems deal with floating values between -1 and 1, implying that are is no higher penalization of a given model feature.

4.2. What is the best regression method to fit the topographic data?

The Ordinary-Leas-Squares method performed significantly better than Ridge and Lasso in terms of MSE. The 3D visualization of the models (Fig. 12) shows that the model predicts most of general terrain features, but the most abrupt topographic variations are generally smoothed-out. For example, the deep river bed located around Y=500 and X= 800 – 1000 (Fig. 2), is totally ignored by the models from our regression analysis. This type of outliers in the data most likely contributed to the increasing test Error (Fig. 10). The Lasso and Ridge regression models performed fairly well in approximating the relief but no better than OLS, both with a similar model error (MSE) and 3D representation.

The reason for the OLS performing better is likely due to the lack of overfitting, as the regularization penalty should not improve the performance of Ridge and Lasso Regression. Ridge also has a higher pernalty than Lasso. Perhaps not significantly higher, since this didn’t seem to have an important effect when comparing the MSE for both methods

In this case, a rescaling of the data improved the performance of the models, due to the different scales between horizontal coordinates (x and y) and the elevation data (z).

To improve these results, we think that a non-linear regression technique, best suited to tackle elevation data, should be tested.

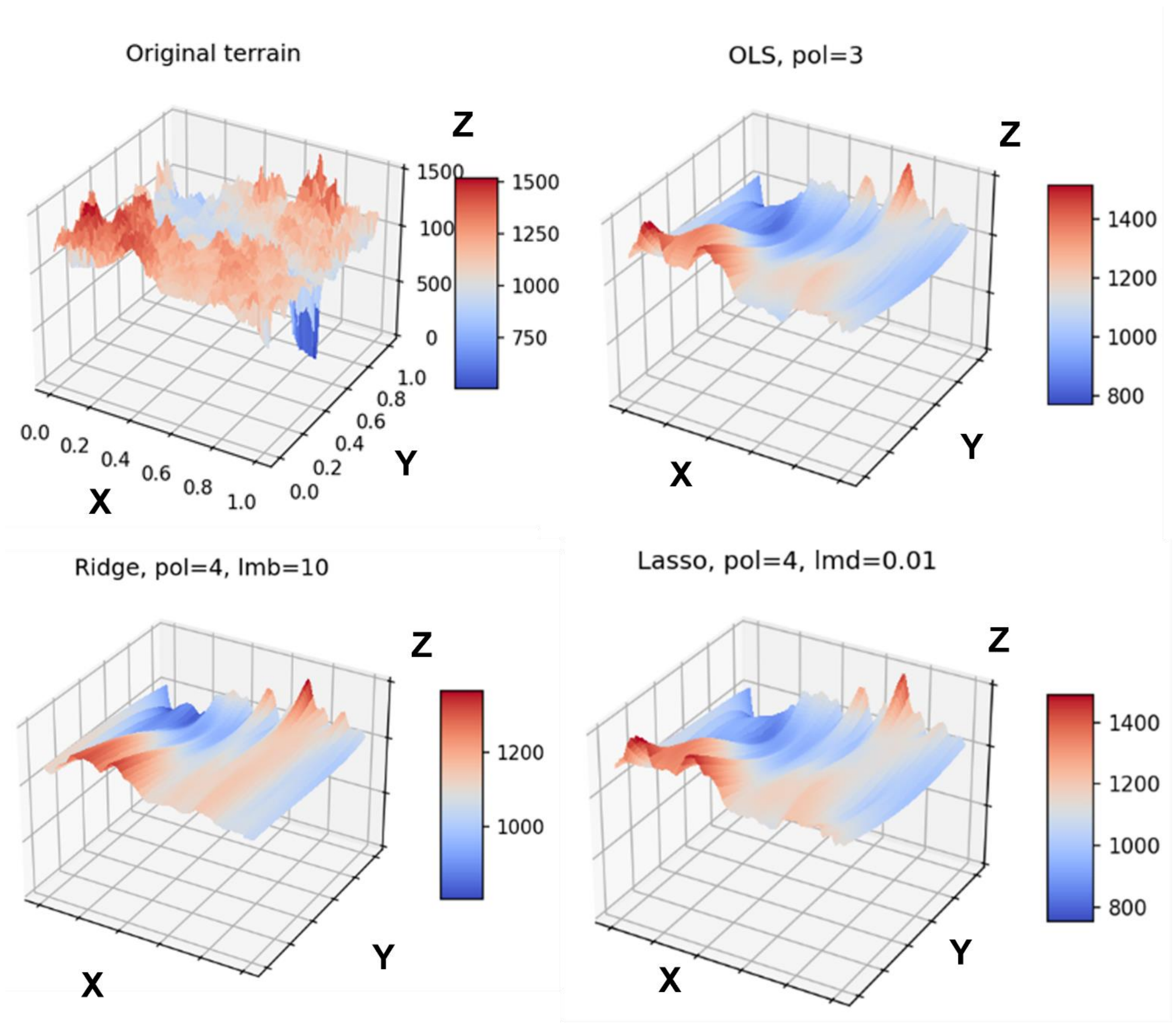


Fig. 12: Visualization of the original terrain data (upper-left), the modeled one with OLS and polynomial degree=3 (upper-right), the Ridge modeled terrain with polynomial degree=4 and λ=10 (lower-left) and the Lasso modeled terrain with polynomial degree 4 and λ = 10^{-1} (lower-right).

5. Appendix

5.1. Source code and test results – GitHub

The python source code and test results are gathered in a GitHub repository:

https://github.com/luisbart/Luis-Romain_STK4155/tree/main/Project1

Program list and brief description:

Functions.py: source code containing functions for the Franke function, MSE, R^2, scaling the data, create design matrix and the 3 types of regression used

01_Plot_FrankeFunction.py: create data, apply Franke function and 3d plot

02_OLS_owncode.py: perform OLS on a synthetic dataset generated by the Franke function. Model evaluation

03_OLS_bootstrap.py: performs bootstrap resampling on OLS regression.

04_OLS_CV.py: performs cross-validation resampling on OLS regression.

05_Ridge_Lasso_MSE_vs_λ.py: performs Ridge and Lasso regression and loop through hyperparameter values. Model evaluation.

06_Ridge_bootstrap.py: performs bootstrap resampling on Ridge regression.

07_Ridge_CrossValidation.py: performs cross-validation resampling on Ridge regression.

08_Lasso_bootstrap.py: performs bootstrap resampling on Lasso regression.

09_Lasso_CrossValidation.py: performs cross-validation resampling on Lasso regression.

10_Terrain.py: performs OLS, Ridge and Lasso on a real terrain elevation data in Norway. Model evaluation by cross-validation

5.2. Supplementary figures

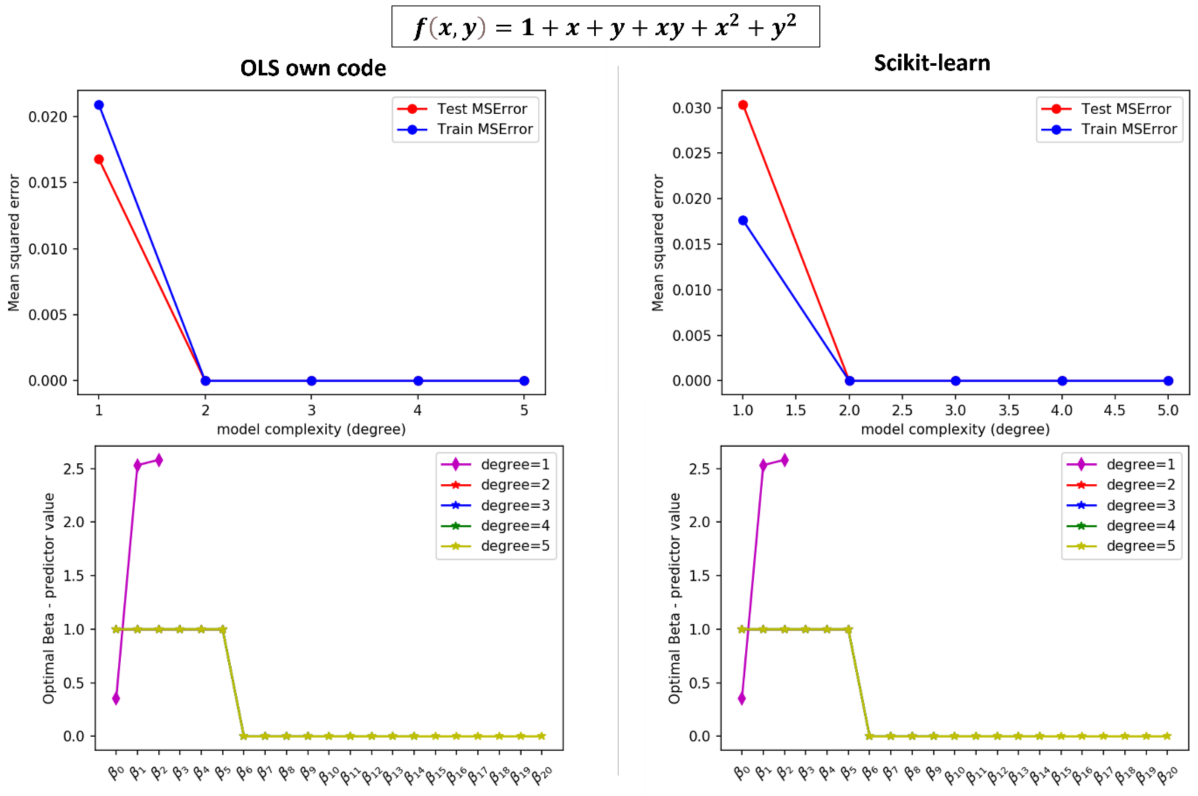


Fig. A1 Regression results of a bi-variate polynomial function f of degree 2 with our own OLS implementation (left panel) and scikit-learn (right panel).

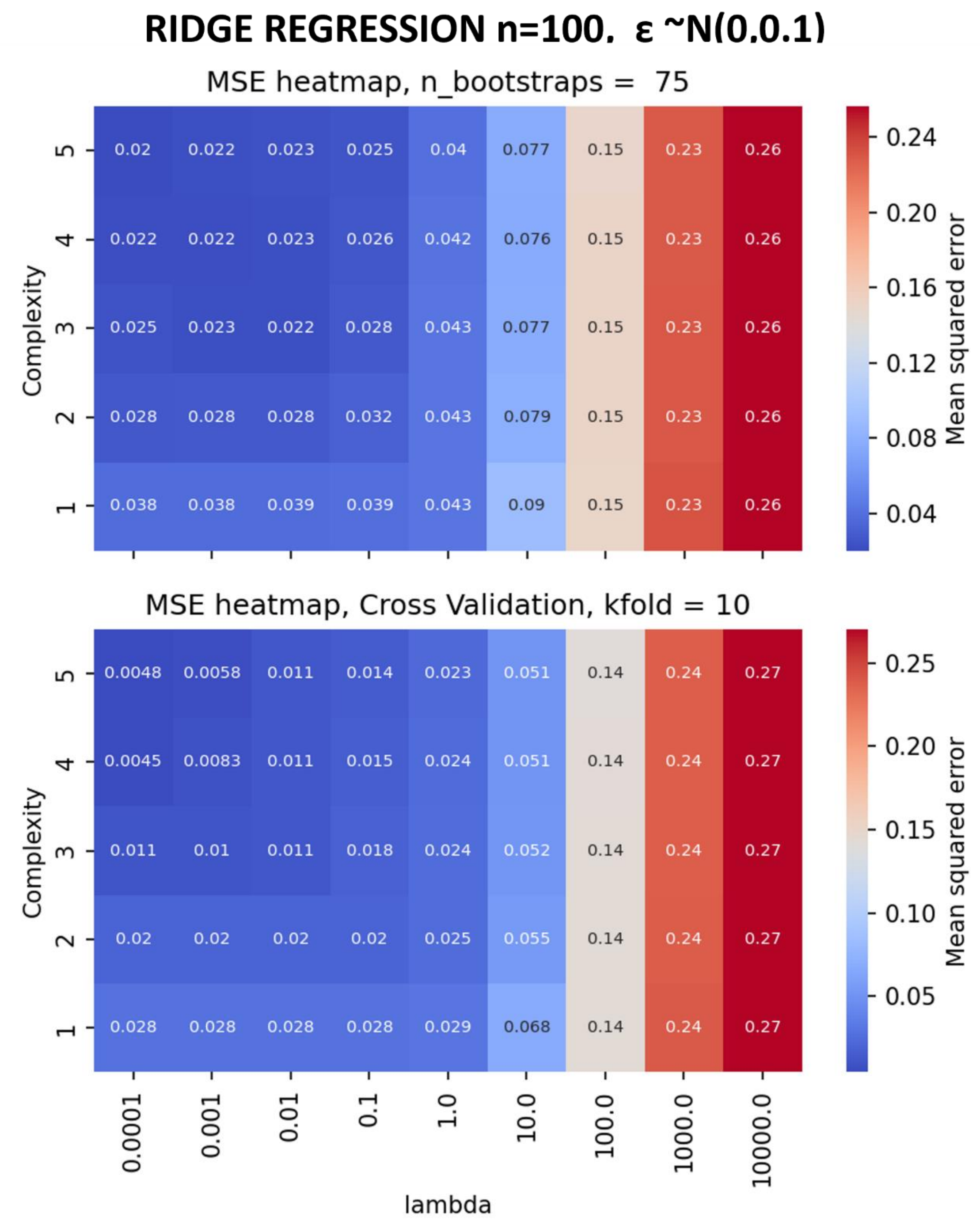
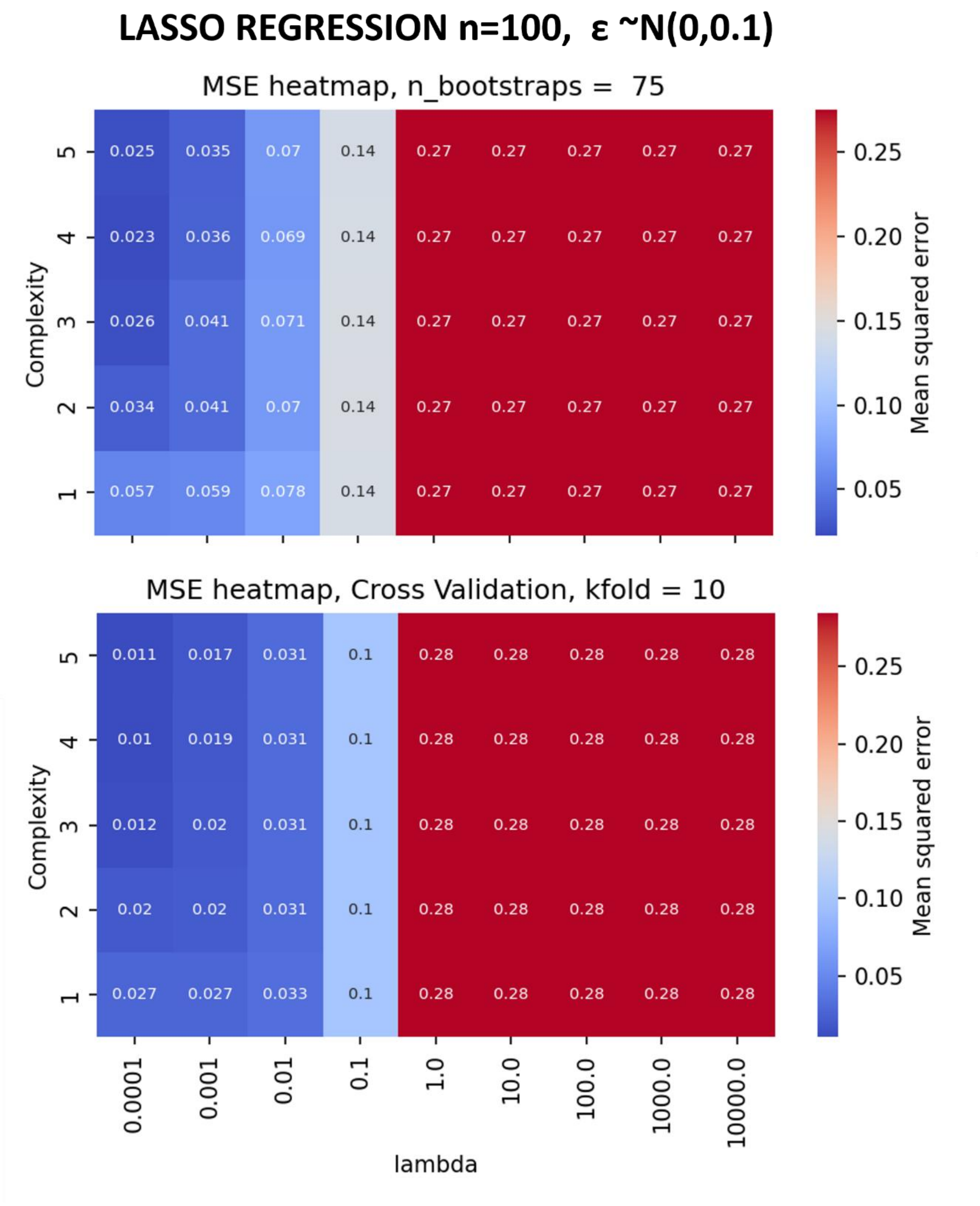


Fig. A2 Heatmaps representing the Ridge regression MSE as function of hyperparameter and polynomial degree using bootstrap (upper panel) and cross-validation (lower panel) re-sampling techniques.



Complexity	0.0001	0.001	0.01	0.1	1.0	10.0	100.0	1000.0	10000.0
5	0.025	0.035	0.07	0.14	0.27	0.27	0.27	0.27	0.27
4	0.023	0.036	0.069	0.14	0.27	0.27	0.27	0.27	0.27
3	0.026	0.041	0.071	0.14	0.27	0.27	0.27	0.27	0.27
2	0.034	0.041	0.07	0.14	0.27	0.27	0.27	0.27	0.27
1	0.057	0.059	0.078	0.14	0.27	0.27	0.27	0.27	0.27

Complexity	0.0001	0.001	0.01	0.1	1.0	10.0	100.0	1000.0	10000.0
5	0.011	0.017	0.031	0.1	0.28	0.28	0.28	0.28	0.28
4	0.01	0.019	0.031	0.1	0.28	0.28	0.28	0.28	0.28
3	0.012	0.02	0.031	0.1	0.28	0.28	0.28	0.28	0.28
2	0.02	0.02	0.031	0.1	0.28	0.28	0.28	0.28	0.28
1	0.027	0.027	0.033	0.1	0.28	0.28	0.28	0.28	0.28

Fig. A3 Heatmaps representing the Lasso regression MSE as function of hyperparameter and polynomial degree using bootstrap (upper panel) and cross-validation (lower panel) re-sampling techniques.

References

Franke, R., (1979). A critical comparison of some methods for interpolation of scattered data. Tech. Rep. NPS-53-79-003, Dept. of Mathematics, Naval Postgraduate School, Monterey, Calif.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., et al., 2020. Array programming with NumPy: Nature, v. 585, no. 7825, p. 357-362 DOI: 10.1038/s41586-020-2649-2.

Hunter, J. D., 2007, MATPLOTLIB: A 2D GRAPHICS ENVIRONMENT: Computing in Science and Engineering, p. 90-95 DOI: 1521-9615/07/$25.00.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., and Thirian, B., 2011. Scikit-learn: Machine Learning in Python: Journal of Machine Learning, v. 12, p. 2825-2830.