A comparative analysis of linear regression methods for predicting terrain data (Project 1)

Ronald Mathew Baysa Payabyab
FYS-STK4155 - Applied Data Analysis and Machine Learning
University of Oslo

October 7, 2020

Abstract

This project aims to study the multivariate linear regression on generic data and a digital elevation model. The three regression models Ordinary least squares, Ridge and Lasso regression were studied further in order to observe their properties, benefits and limitations. Fundamental concepts of machine learning methods are being emphasized in this project, hereby bias-variance tradoff and resampling techniques. All three regression methods predicted both the given data sets with relatively proper results. OLS and to some extent Ridge in particular appears to overfit the datasets, which seems to less of a problem with larger sets of data. Lasso on the other hand did not have the same issue, but the resistance to overfitting seems to be at the expense of runtime performance. When applying regression analysis with some stochastic noise $\sigma = 0.1$, the maximum R2-score peaked at 0.9269 at the 5th degree polynomial with $MSE = 0.005$ when using OLS. OLS also did perform better than the other regression methods for the terrain data as well, giving an R2-score of 0.774 and a MSE of 0.0283 for the 10th degree polynomial. An optimal polynomial degree was however difficult to determine as it potentially is substantially large for such complex systems as the terrain data.

1 Introduction

One of the advantages of utilizing regression analysis is due to its simplistic nature compared to its more complex counterparts. Linear regression is also more interpretable as it shows the linear relationship between the independent and dependent variables in addition to usually tend to be less computationally expensive. This project will take in use linear regression as a machine learning algorithm to explore its ability to predict generic data and real terrain data. Most part of this project will emphasize on studying and observing properties of machine

learning algorithms such as resampling, bias-variance, and underfitting/overfitting of data, which is recurrent in all machine learning techniques.

The data sets for this project contains two exploratory variables, and the three regresion methods ordinary least squares (OLS), Ridge regression, and Lasso regression will be employed upon those. The first data set is generated from the Franke function with some stochastic noise added. The second data set contains a digital terrain model of western Hokkaido, Japan for the purpose of studying a much larger and more complex set of data to apply to the regression methods.

2 Background

Linear regression describes a sampling distribution of a given response variable y. The model assumes that the regression function $E(Y|X)$ is linear to input data $X_1, X_2, ..., X_p$, where X is given as a variable or a set of variables called the predictor variable. In other words, a linear model makes a prediction by computing a weighted sum of input features including a constant called the bias term β_0

$$\hat{y}_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2k} + ... + \beta_k x_{ki} + \epsilon_i \quad , (i = 1, ..., n) \tag{1}$$

where k is the number of features in the data set, $X_k i$ is the ki-th feature value, β_j is the j-th model parameter including the bias term β_0 and its feature weights $\beta_1, ..., \beta_k$, and ϵ_i represents some added noise to the data. For analytical purposes, it is convenient to express the model (1) in matrix form:

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{21} & \dots & x_{k1} \\ \vdots & \vdots & & \vdots \\ 1x_{21} & x_{2n} & \dots & x_{kn} \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}, \epsilon = \begin{pmatrix} \epsilon_0 \\ \vdots \\ \epsilon_k \end{pmatrix} \tag{2}$$

Here, the nxk-matrix represents the design matrix $X = (x_{ij}$, where the first index j ($j = 1, ..., k$) refers to the variable number in columns, while the second index i ($i = 1, ..., n$) refers to the observation number in rows. The notation in (2) can further be rewritten as:

$$y = \hat{y} + \epsilon = X\beta + \epsilon \tag{3}$$

β is a $k \times 1$ vector of unknown parameters, while ϵ is an $n \times 1$ vector of unobserved disturbances. $\hat{y} = X\beta$ on the other hand represents the predicted value in the data set.

2.1 Ordinary Least Squares

It is usual that the analysis contains a set of training data $(x_1, y_1), ..., (x_N, y_N)$ to estimate the confidence intervals β. Each $x_i = (x_{i1}, x_{i2}, ..., x_{ip})^T$ is a vector of measurements for the

i-th case Hastie et al. (2001). The most common methods in regression analysis is the least squares, where the β-coefficients are minimized to the residual sum of squares:

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{4}$$

The equation does not make any assumptions about the model validity, but rather finds the best linear fit to the data. To minimize the equation, it is possible to rewrite the residual sum-of-squares through denoting the $N \times (p+1)$ matrix by X with each row an input vector, and letting y be the N-vector of outputs in the training set:

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \tag{5}$$

When differentiating with respect to β, the expression can be rewritten further as:

$$\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \tag{6}$$

$$\frac{\partial^2 RSS}{\partial \beta \partial \beta^T} = 2\mathbf{X}^T\mathbf{X} \tag{7}$$

The first derivative of the equation can only be set to zero if it is assumed that $\mathbf{X}$ has full column rank, which leads to $\mathbf{X}^T\mathbf{X}$ being positive definite, and thus giving a unique solution:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \tag{8}$$

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{9}$$

The β-parameters in ordinary least squares minimizes the difference between the target vector y_i and the prediction $\hat{y}_i$. In the linear regression model, training a model means setting the parameters in a way that the model best fits the training set.

The approximation $\tilde{y} = \mathbf{X}\boldsymbol{\beta}$ is defined by the unknown $\boldsymbol{\beta}$, and in order to find its optimal parameters a function is defined by giving measure of the spread between the exact values y_i and the parameterized values $\tilde{y}_i$. Using the design matrix, this represents the cost function:

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\} \tag{10}$$

To find the β-parameters, it is necessary to minimize the spread of the cost function, which in other means that it is required to solve the problem:

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{11}$$

which can further be written as:

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \tag{12}$$

Considering that $\mathbf{X}^T\mathbf{X}$ is invertible, one can obtain the solution:

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{y} \tag{13}$$

2.2 Ridge regression

Ridge regression is a slight modification of the ordinary least squares. It is a regularized term of linear regression with a defined cost function of

$$C^{Ridge}(\beta) = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - \boldsymbol{X_i}\boldsymbol{\beta}|^2 + \lambda \sum_{j=0}^{p-1} \beta_j^2 \tag{14}$$

The cost function of Ridge is similar to ordinary least squares with an added term that represents the size of β-vectors, multiplied by a hyperparameter λ. The hyperparameter controls how much the model should be regularized. If $\lambda = 0$, then the Ridge regression is just a linear regression. On the other hand, if λ is sufficiently large, then all weights end up very close to zero and the result is a flat line going through the data's mean values. The main purpose of Ridge regression is to penalize the slopes $\beta_i, \neq 0$ of the model, which is proportional to the hyperparameter, and thus are expected to have lower coefficient values than the ordinary least squares. Increasing the hyperparameter values can be interpreted as being less confident about the input data's ability to predict the model output, as the β-coefficient values define the correlation between the input and output data. Thus, if the minimum of the cost function gets steeper around a β-value, the less it will by Ridge, implying that less confident values get moved more. The analytical expression of β is similar to ordinary least squares, and is also derived from the cost function:

$$\beta = (\boldsymbol{X^T X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X y} \tag{15}$$

with the hyperparameter added along the invert of the diagonal matrix. Adding the hyperparameter will also make the diagonal matrix non-singular if the diagonal matrix was singular prior to adding a small valued hyperparameter.

2.3 Lasso

Lasso regression is similar to Ridge regression, with its cost function defined as:

$$C^{Lasso} = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - \boldsymbol{X_{i*}}\boldsymbol{\beta}|^2 + \lambda \sum_{j=0}^{p-1} |\beta_j| \tag{16}$$

An important characteristic of Lasso compared to Ridge is that it completely eliminates the weights of the least important features, by suppressing some of the β-coefficients to zero. Since β is squared for Ridge regression, the penalty of having a small coefficient value is small. This is on the other hand not the case for lasso regression. as reducing some coefficient to zero might give a prediction to the cost function.

2.4 Singular value decomposition

It is however not guaranteed that $\mathbf{X}^T\mathbf{X}$ is invertible. A standard matrix inversion may lead to singularities. A "quick fix" of this issue can be resolved by estimating the inverse of the matrix. The singular value matrix decomposition (SVD) makes it easier to compute the inverse of a matrix, allowing any matrix A to be in the form of:

$$A = UDV^T \tag{17}$$

where U and V are orthogonal matrices whose columns form an orthonormal basis and D is the diagonal matrix where the only non-zero values lie along the diagonal representing the singular values of matrix A:

$$D = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_p \end{bmatrix} \qquad , \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0.$$

The SVD exists for any matrix, and can be uitilized without any assumptions on the target matrix, meaning that any linear function can be written as a fixed scaling of the components and another rotation. It exploits the fact that U and V are orthogonal such that $U^TU = UU^T = I$ and $V^TV = VV^T = I$. Inserting SVD to the ordinary least squares gives:

$$A^{-1} = VD^{-1}U^T \tag{18}$$

It is possible to find the inverse by undoing each of three operations when multiplying A. First, we undo the last rotation by multiplying with U^T. Secondly we unstretch by multiplying the inverse of the diagonal matrix, and then unrotate by multiplying with V (source). In other words, the definition is correct by proving that A is orthogonal:

$$A^{-1}A = (VD^{-1}U^T)(UDV^T) = VD^{-1}(U^TU)DV^T = V(D^{-1}D)V^T = VV^T = I \tag{19}$$

2.5 Variance and confidence intervals

After gaining the values of the β-coefficients it is important to check their level of confidence as they are subject to sampling uncertainty. In other words, it impossible to exactly estimate the true value of these parameters from sample data. To obtain the sampling properties of $\hat{\beta}$, we assume that y_i are uncorrelated with a constant variance σ^2 with x_i fixed points. According to Hastie et al. (2001), the variance-covariance matrix with least squares parameters could defined as:

$$Var(\hat{\beta}) = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2 \tag{20}$$

where σ^2 is usually estimated by:

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{21}$$

Here, the denominator is defined as $N-p-1$ rather than N to ensure that $\hat{\sigma}^2$ is an unbiased estimate of σ^2.

Since it is known that the multiple linear regression model $\hat{\beta}$ is a multivariate normal and covariance matrix as defined by equation (21), we can deduce the confidence interval. Given $0 \leq i \leq k$, $\hat{\beta}_i$ is normally distributed with mean β_i and variance $\sigma\sqrt{Cii}$ if C_{ii} is the i-th diagonal entry of $(\mathbf{X}^T\mathbf{X})$, the $(1-\alpha) \times 100\%$ confidence interval for β_i is then given by:

$$\hat{\beta}_i \pm z^{1-\alpha}\hat{\sigma}\sqrt{C_{ii}}, \quad \forall \quad 0 \leq k \tag{22}$$

where α describes the slope and $z^{1-\alpha}$ represents the $1-\alpha$ percentile pf the normal distribution (Hastie et al., 2001).

2.6 Error analysis

When performing machine learning analysis on data, it is typical to provide a performance measure for regression problems as it gives an idea of how much error the system makes in its predictions, with a higher weight for large errors. The equation below shows a mathematical formula of how to compute the mean squared error (MSE):

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y})^2 \tag{23}$$

The MSE provides a description of the squared difference between the predicted data and true data. This is also defined as the cost function the ordinary least square utilizes, meaning that the MSE minimizes the data with respect to the data it was trained on.

Another common metric used in performance measurement is the coefficient of determination (R2 score), expressed as:

$$R2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{24}$$

The R2 score gives an idea of how many data points fall within the results of the line formed by the regression equation. The higher score, the higher the percentage of points the line passes through the data points, which indicates that there is a better goodness of fit for the observations. The R2-score is useful to find the likelihood of future events falling within the predicted outcomes.

2.7 Bias-variance tradeoff

The model's generalization error is regarded as crucial for analyzing the dataset in machine learning. The generalization error can be expressed as the sum of the errors: bias, variance

and irreducible error (Géron, 2017). The bias represents the part of the generalization error due to wrong assumptions, for instances when assuming the data is linear when it is actually quadratic.

$$bias[\tilde{\mathbf{y}}] = \mathbb{E}[\tilde{\mathbf{y}}] - f \tag{25}$$

Note that the bias is defined by the difference of the average value of predictions over different realizations of training data to the true underlying function f for a given unseen point x (Papachristoudis, 2019). This means that we have access to only a portion of underlying data as training data. When the expectation $\mathbb{E}[\tilde{\mathbf{y}}]$ is over different realization of training data, it simply means that the algorithm polls a sample out of the underlying data multiple times and gets the average values of the predicted data. Thus, $\tilde{y}$ changes even if x is fixed. When the model complexity is low, the fit might miss features in the training data set. Once the model is applied to the test data, the expectation value $\mathbb{E}(\tilde{y})$ will generate substantially different values than the actual data. The bias is considered high, and the model is underfitting.

The variance on the other hand expresses the model's sensitivity to small variations in the training data. A model with many degrees of freedom (e.g the model complexity is large) is more likely to have high variance, since the fit is more likely to predict detailed features in the training set in addition to consistencies generated by noise. Hence, the training data is being overfitted. The variance can be expressed as:

$$Var(\tilde{\mathbf{y}}) = \mathbb{E}[(\tilde{\mathbf{y}} - \mathbb{E}[\tilde{\mathbf{y}}])^2] \tag{26}$$

The irreducible noise is the part due to the noisiness of the data itself, and the only way to reduce this error is by cleaning up the data (e.g fixing data points, removal of outliers etc.). In general increasing the the model complexity will usually increase the variance while reducing the bias at the same time. Conversely, reducing the model complexity increases its bias while reducing its variance. When resampling the data with multiple iterations, the predicted data tends to be varying, and thus leads to higher variance.

To show the connection between test MSE and bias-variance, assume that we have the underlying data points y, and ϵ represents the noise in the data. Additionally the test MSE, $\mathbb{E}[(y - \tilde{y})^2]$, is over the realizations of training data, which takes the form of:

$$\begin{aligned} \mathbb{E}[(y - \tilde{y})^2] &= \mathbb{E}[(f + \epsilon + \tilde{y})^2] \\ &= \mathbb{E}[(y - \tilde{y})^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(y - \tilde{y})\epsilon] \\ &= \mathbb{E}[(y - \tilde{y})^2] + \sigma_\epsilon^2 \end{aligned} \tag{27}$$

In the equation, y has the input $y = f + \epsilon$. Noise is modeled by the random variable ϵ with zero mean and variance σ_ϵ^2. The magnitude of variance shows the uncertainty levels about an underlying phenomenon. The variance σ_ϵ^2 gets larger when the uncertainty increases. Mathematically, ϵ has the following properties from equation (28):

$$\mathbb{E}[\epsilon] = 0, Var(\epsilon) = \mathbb{E}[\epsilon^2] = \sigma_\epsilon^2 \tag{28}$$

The product of equation (28) shows how the test MSE decomposes to the irreducible error σ_ϵ^2 and $\mathbb{E}[(y-\tilde{y})^2]$. By analyzing the latter term further, we get:

$$\begin{aligned}\mathbb{E}[(f-\tilde{y})^2] &= \mathbb{E}[((f-\mathbb{E}[\tilde{y}]-(\tilde{y}-\mathbb{E}[\tilde{y}]))^2] \\ &= \mathbb{E}[(\mathbb{E}[\tilde{y}]-f)^2]+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2]-2\mathbb{E}[(f-\mathbb{E}[\tilde{y}])(\tilde{y}-\mathbb{E}[\tilde{y}])] \\ &= (\mathbb{E}[\tilde{y}]-f)^2+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2]-2(f-\mathbb{E}[\tilde{y}])(\mathbb{E}[\tilde{y}]-\mathbb{E}[\tilde{y}]) \\ &= bias(\tilde{y})^2+Var(\tilde{y})\end{aligned} \tag{29}$$

The last term in the equation cancels out, and accordingly to the equations (25) and (26), we are left with the $bias^2$ and the variance. By adding the noise, this will yield.

$$\mathbb{E}[(f-\tilde{y})^2] = bias(\tilde{y})^2+Var(\tilde{y})+\sigma_\epsilon^2 \tag{30}$$

By inserting the cost function, equation (22) can be rewritten as:

$$\mathbb{E}[(y-\tilde{y})^2] = \frac{1}{n}\sum_i (f_i-\mathbb{E}[\tilde{y}])^2+\frac{1}{n}\sum_i(\tilde{y}_i-\mathbb{E}[\tilde{y}])^2+\sigma^2 \tag{31}$$

2.8 Resampling techniques

One of the lacklusters of machine learning methods is the lack of data. Data is crucial to train the model, test the validity, and inspect different properties such as the bias and variance. Resampling techniques is therefore applied to split, reuse, and improve the data in different ways to get the most out of it.

2.8.1 Bootstrap

Bootstrap sampling helps avoiding overfitting the data and improves the stability of machine learning algorithms. When aggregating using bootstrap, a certain number of equally sized subsets of a dataset is applied to each of these subsets. It allows training instances to be sampled several times across multiple predictors. Once all the predictors are trained, the ensemble can make a prediction for a new instance by aggregating the predictions of the predictors.

2.8.2 Kfold Cross validation

Cross validation works as a great alternative for residual evaluation. The issues with residual evaluations is that they do not give an indication of how well the learner will do when asked to make predictions for data it has not already seen. The basic principle for cross validation is to not use the entire data set when training a learner, by removing some of the data before

the training begins. When the training is finished, the data that was removed can be used to test the performance of the learned model on new data.

A commonly used cross validation technique is the kFold. The data is divided into k subsets, and the holdout method is repeated k times. One of the k subsets is used as the test set, while the rest of the k-1 sets are put together to form a training set. The average error across all k trials are then computed. An advantage of using this method is that it matters less how the data gets divided since every data point gets to be in a test set at least once, and to be in a training set k-1 times. The variance of the resulting estimate is reduced as k increases.

3 Methodology

3.1 Franke's Function

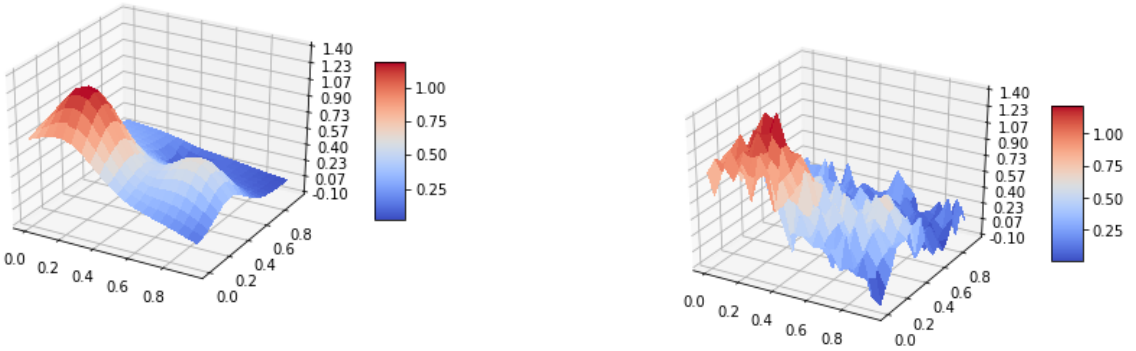
As an example for implementing the different types regression analyses, we first apply some generic data. The data will be generated from the given Franke's function, which is a smooth exponential function used to test interpolation methods. The function is given as:

$$\begin{aligned} f(x,y) = \frac{3}{4}exp(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}) + \frac{3}{4}exp(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}) \\ + \frac{1}{2}exp(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}) - \frac{1}{5}exp(-(9x-4)^2 - (9y-7)^2) \end{aligned} \tag{32}$$

Some Gaussian noise $\sigma = 0.1$ was added in the function in order to represent a more realistic real world data. The function used for generating data in the analysis is thus expressed as:

$$f(x,y,\sigma) = f(x,y) + N(0,1), \quad \forall \quad x,y \in [0,1] \tag{33}$$

A mesh of 200 points were used in both the x, and y-axis, which gives 40 000 data points across the mesh. The surface plot will then experience some alterations when applying noise to the data.



(a) Without noise

(b) With noise, $\sigma = 0.1$

Figure 1: Franke's function

3.2 Terrain data: Hokkaido, Japan

For implementing the analysis with real world data, a digital elevation model (DEM) of Western Hokkaido, Japan was downloaded from **earthexplorer.usgs.gov**. The variables used in the analysis remains the same as for the Franke's function. The size of the DEM is rather large. A 3601×3601 DEM image, gives 12967201 data points/pixel values in total. In order to hamper the issues with runtimes, it was essential to downsample and resize the image by 15×15 pixels, which will approximately generate $241 \times 241 = 58081$ data points. In addition to the x- and y-coordinates, the elevation represents the z-coordinates given in meters.

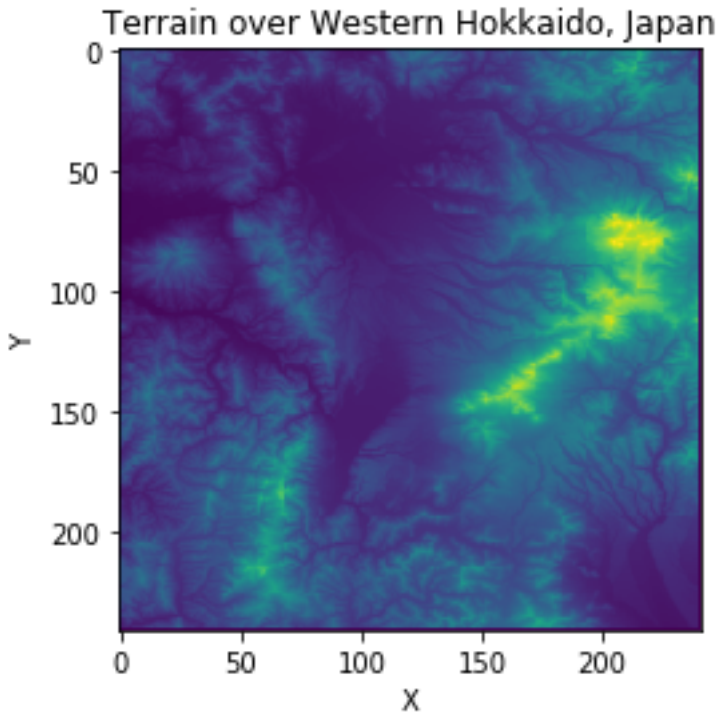


Figure 2: Terrain plot over the study area

3.3 Code implementation

The analysis stems from 5 different Python-files. The methods of linear regression were applied as a class function in **Regress.py**. The methods of OLS, Ridge, and Lasso and their respective results of β-coefficients can be found there, including some algorithm for fitting data accordingly to each method, and finding prediction values. It also includes a function to change the lambda-values for Ridge and Lasso. It should be noted that the methods for ordinary least squares and Ridge were coded manually, while the methods of Lasso regression were implemented by the help of functionalities in scikit-learn.

Resampling techniques such as the Kfold cross validation and bootstrap are stored in another class function named **Resampling.py**. This is also where most of the error scores R2, MSE, bias and variance are defined and generated. In addition to the two resampling techniques, the class function also provides the option to generate error values without resampling the data. Furthermore, most of the functionalities for plotting error analysis, learning curve, confidence intervals, learning curves, and bias-variance tradeoff are stored in **results.py**. Other kinds of functions, such as generating the design matrix, the Franke Function, plotting terrain data, and creating the inverse function by singular value decomposition can be found in **list_of_func.py**. Finally, a **main.py** file was created for the purpose of calling all the algorithms to generate plots and printing out results of the error analysis.

3.4 Multivariate polynomial regression

Both the Franke function and the terrain data for Hokkaido, Japan are defined with one set of output data of a function, which are predicted as a polynomial of two sets of input data by the variables of **x** and **y**. By inserting the input variables in the function, the function will then become a two-dimensional surface. For instances, a second order multiple polynomial regression will then be represented as:

$$z = \beta_0 + \beta_1 x + \beta_2 y + \beta_{11} x^2 + \beta_{22} y^2 + \beta_{12} xy \tag{34}$$

This will also define a response surface which can be represented in a matrix form as usual:

$$Y = \beta \mathbf{X} + \epsilon \tag{35}$$

It will still fit in a polynomial regardless, creating a smooth two-dimensional surface. Here, **X** will take the form of n x p - matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & x & y & xy & \dots & xy^{m-1} & x^m & y^m \end{bmatrix}$$

where m is defined as the polynomial degree. Since cross term relations occur, p is defined as $\frac{(m+1)(m+2)}{2}$ which also represents the length of the β.

3.5 The analysis

The analysis was first performed as regular OLS regression as a 10th order polynomial without bootstrap or cross-validation in order to verify if every step in the analysis goes as expected. It should be noted that it is preferable to always do a cross validation as the risk of overfitting data is higher and it gives limited options for measuring the performance without such implementations. A noise of $\sigma = 0.1$ was added throughout the entire analysis, but an OLS-analysis without noise were also investigated upon.

The data sets used for training and testing was centered and scaled prior to performing resampling techniques using Scikit-learn's StandardScaler to prevent any outliers to intervene the data sets. It standardizes features of the dataset by removing the mean and scaling to unit variance. Centering and scaling happen idependently on each feature, and the mean and standard deviation are stored to be used on later data for the Transform-function in Scikit-learn. This is were the standardization happens by centering and scaling.

Checking the error metrics MSE and R2 as discussed in 2.9 defines the core of the regression analysis. The MSE and R2 was applied manually by the definitions of equation (10), while the MSE and R2 with bootstrap uses the built-in functions in scikit-learn's **sklearn.errormetrics**. Additionally, the variance and the β-coefficient's confidence intervals was also taken into consideration. This step accounts for both the Franke function and the terrain data set.

3.6 Bootstrap and Kfold cross validation

The next step further in the analysis is to apply some resampling with bootstrap and Kfold-cross validation. In the analysis, 100 bootstraps was implemented by default for each regression method. In Kfold cross validation, the partitioning of data into k number of folds were performed manually. It is defined by a function that generates "k" (training,testing)-pairs from the items in the design matrix X. Each pair is a partition of X, where the test set is an iterable of len(X)/K. It also gives and opportunity to randomize the data. When randomized, a copy of X is shuffled before partitioning. The default number of folds was set to 5 folds.

The error metrics were analyzed and compared to the results from the ordinary least squares prior to applying bootstrap or kFold cross validation for both Franke function and the terrain data. It was expected that overfitting would disappear as the algorithm now take use of different parts of data sets.

3.7 Ridge and Lasso

After studying the error metrics of ordinary least squares, the analysis moved further towards the regression methods Ridge and Lasso. The β-coefficients has an analytical solution and were thus applied manually defined as mentioned by equation (19). The Ridge regression added and extra parameter that needs optimization. It is thus necessary to investigate a proper λ-value as it determines the model performance. In order to study the influence of the λ-parameter on the error metrics, several Ridge regression were performed over different λ-values spanning in the interval $[10^{-10}, 10^2]$ for both the Franke function and terrain data. A plot with the results of each regression analysis against the corresponding λ-values were compared to the error metrics from the standard ordinary least squares analysis.

Unfortunately, Lasso does not have an analytical solution for β. To make the the process easier, the algorithm for finding the β-coefficients was borrowed from Scikit-learn's packages **sklearn.linear_model.Lasso**. The same steps was performed in Lasso as of Ridge regression, but since Lasso does not have an analytical solution for minimizing the cost function, the computation time took remerkably longer compared to Ridge. Thus, it was necessary to study a smaller interval for λ.

3.8 Bias-variance tradeoff

A high variance often leads to overfitting a model, which will be troublesome for more complex models. The training and test MSE of the model was therefore studied along with increasing model complexity, which in this case represents the polynomial degrees of freedom.

By default, the dataset were splitted for 80% training and 20% testing. The model was run from the polynomial order in the interval $[1, 10]$. The λ-value will then be a fixed value $\lambda = 10^{-5}$.

For calculating the bias and variance, the Bootstrap method was used which contained 80% training data and 20% test data. The bootstrap is only implemented on the training data, keeping the test data in a vault. The analysis will then leave behind several different fits when ran with n different bootstraps, which are used to calculate all the values involved in the bias and variance terms of MSE. The bootstrap was performed for all three regression methods and for both data sets involved in the analysis. To study the connection between MSE, model complexity and λ-parameters even further, an attempt to create a heat map was made using the Python library **Altair**.

3.9 Learning curves

It is known that cross-validation estimates the model's generalization performance. A model is overfitting when it generalizes poorly according to cross-validation metrics even if it performs well on the training data. If the model also perform badly for training data as well, then it is underfitting.

An alternative way to study generalization is by creating a learning curve. A learning curve plots the model performance based on the training set and the test set as a function of the training iteration. When creating the learning curve, the model is trained several time on different subsets of the training set, which have different sizes each time. The learning curve was created for all three regression methods for both data sets by using bootstrap in the Franke function, and Kfold cross validation for the terrain data.

4 Results and discussions

4.1 Ordinary least squares

Table 1: The ranking process of each classses for each criterion

Ordinary least squares				
Resampling method	**MSE**	**R2 score**	**Optimal degree**	**Applied noise?**
Train test	0.0037	0.9442	6.0	Yes
Bootstrap	0.005	0.9269	5.0	Yes
KFold	0.004	0.955	5.0	Yes
Train test	0.0001	0.998	6.0	No
Bootstrap	0.001	0.9865	8.0	No
KFold	0.0003	0.9958	10.0	No

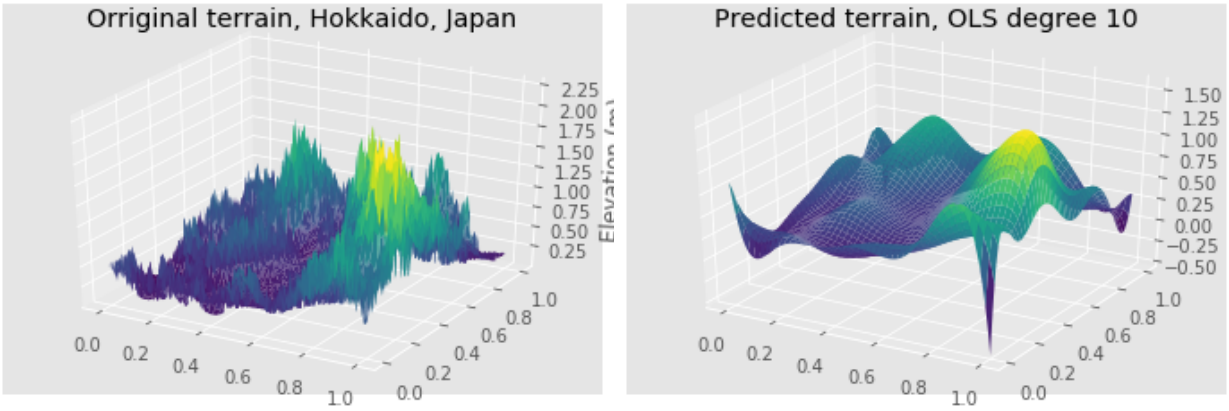
The table above shows the error metrics of performing an ordinary least squares regression on the Franke function by different resampling methods. As previously mentioned, the noise was set to $\sigma = 0.1$. It could be observed that adding stochastic noise slightly alters the error values, as the R2 score almost reaches a value of 1.0 regardless of resampling method chosen when not applying any noise. It is however notable that the optimal degree for the noiseless function are slightly higher than the Franke function with added noise.

The terrain data on the other hand, has much worse fit than the Franke function for all resampling methods as shown in table 2, when attempting a 5-fold cross validation up to a polynomial degree of 10. Given that the R2-score dropped significantly in comparison to the R2-score in the Franke-function, a much higher order polynomial order is needed to fit the dataset in the elevation model better.

Table 2: OLS error metrics from terrain data at 10th degree polynomial

Ordinary least squares		
Resampling method	**MSE**	**R2 score**
Train test	0.0289	0.7757
Bootstrap	0.0284	0.7052
KFold	0.0283	0.774

From figure 3, it is observed that the predicted terrain of a 10th polynomial order generates a much smoother terrain in comparison to the original terrain, which is more rugged. It also observed that the OLS-prediction has some obvious mispredictions as well, as some of the elevation values reaches below sea level.

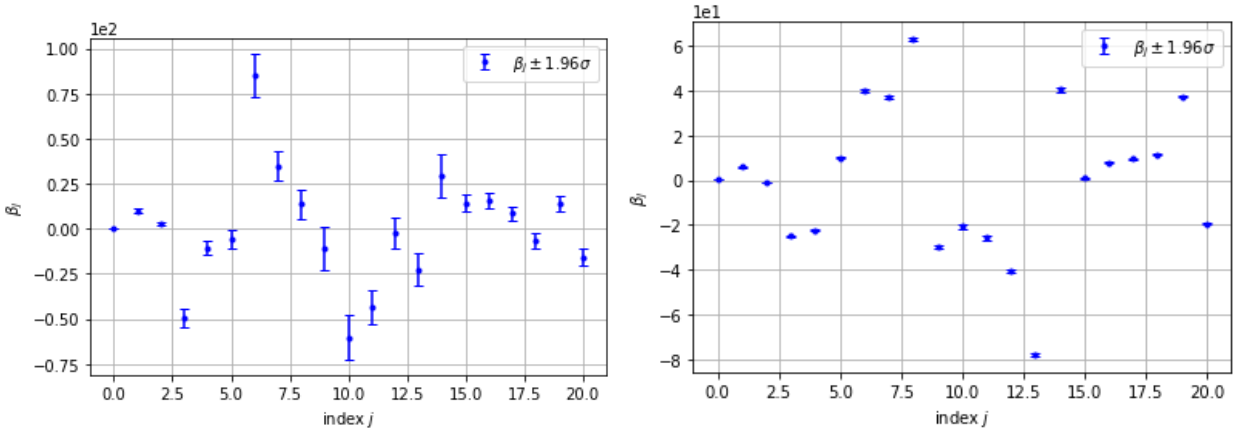


(a) Original terrain data

(b) Predicted terrain

Figure 3: DEM of Hokkaido, Japan. Left image shows the original terrain while the right image shows the predicted terrain in 10th-degree OLS with 5-folded cross validation.

4.2 Confidence intervals



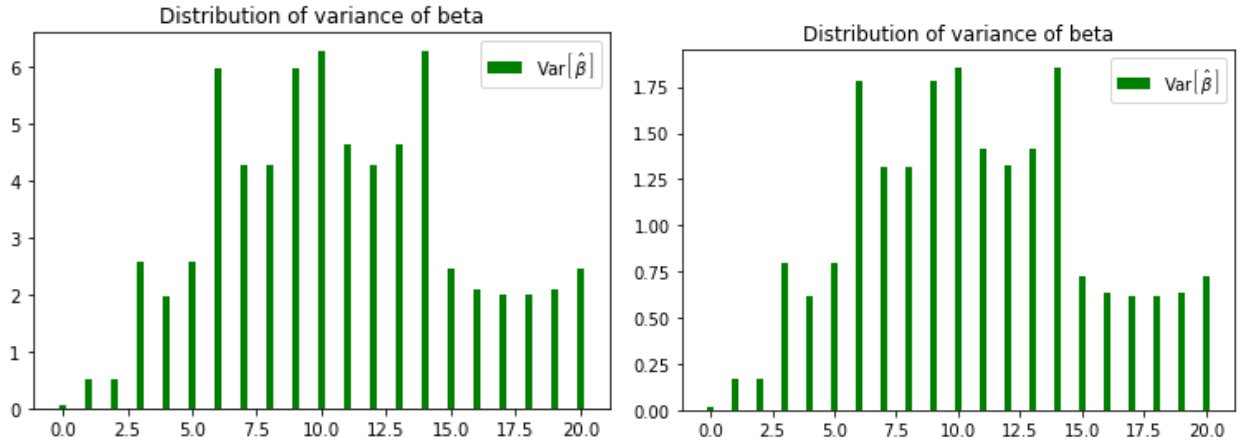
(a) Franke's function

(b) Terrain data

Figure 4: Confidence intervals using OLS up to 5th degree polynomial

The figures above illustrates the predicted values of the β-coefficients of Franke's function and the terrain data with 95% confidence. The confidence intervals are much smaller for the terrain data in comparison to the Franke function. The confidence intervals of the Franke function tends to have small values for the first three β-coefficients. Furthermore, the parameters between $j = 5$ and $j = 15$ appears to have larger confidence intervals while the higher and lower indices tend to have the opposite, which might indicate that the predictors these coefficients represent do not affect the model too much.

Judging from the confidence interval of the terrain data at figure 4b, it appears that all of the β-coefficients and their respective predictors do not affect the model too much or at all, since the confidence intervals are very small if it exist at all. This is something not to be expected since the OLS model did a slightly worse fit to the terrain data compared to Franke's function.



(a) Franke function (b) Terrain data

Figure 5: OLS bootstrap

Additionally, a histogram showing the distribution of the $var(\beta)$ was plotted accordingly to the discussion in section 2.5. Interestingly enough, the histograms of both the datasets follows to the same pattern, although the values of the β-coefficients are different. On the other hand, the x- and y-axis are spaced in the same interval $x, y \in [0, 1]$ for both datasets. Their desingn matrices should thus be similar, which also yields identical values for $X^T X$, as $X^T X$. It also observed that β_0 has notable smaller variance when compared to other values in both the datasets.

4.3 Ridge, Lasso and the hyperparameter

Table 3: The ranking process of each classses for each criterion

Ridge regression for $\lambda = 10^{-5}$				
Resampling method	**MSE**	**R2 score**	**Optimal degree**	**Applied noise?**
Bootstrap	0.0048	0.9128	9.0	Yes
Kfold	0.0045	0.95	10.0	Yes

The table above shows the optimal error values for Ridge regression with a given hyperparameter $\lambda = 10^{-5}$. Both the R2-score and the MSE values are slightly worse compared to

the results of the OLS. The optimal degrees are also significantly higher for both resampling methods. Since the complexity are only analyzed up to 10th degree polynomial, a higher degree polynomial might be considered for fitting the model.

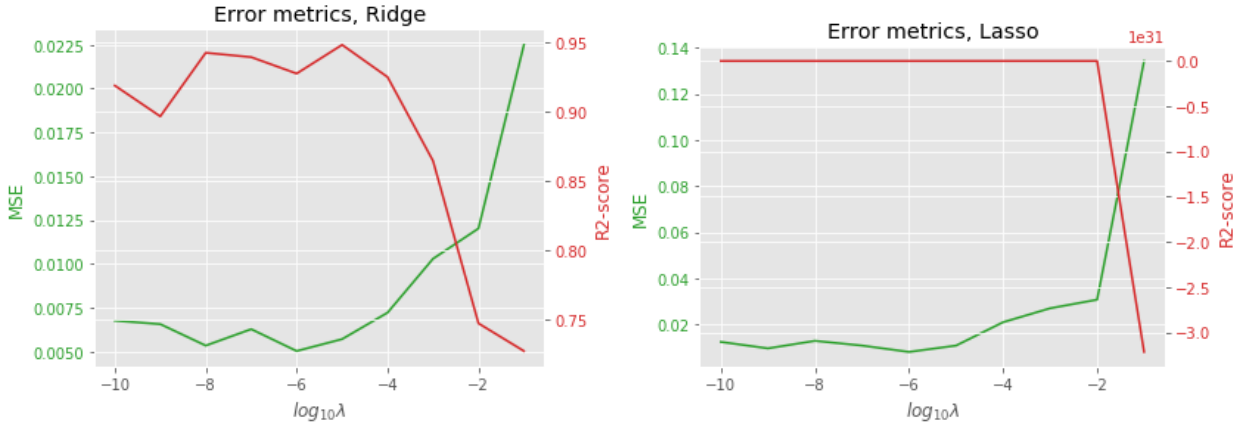
Similarly to OLS, the terrain data did give slightly worse results as well, with $MSE = 0.034$ and $R2 = 0.7285$ for the 10th-degree polynomial using Kfold-cross validation.

Table 4: The ranking process of each classses for each criterion

Lasso for $\lambda = 10^{-5}$				
Resampling method	**MSE**	**R2 score**	**Optimal degree**	**Applied noise?**
Bootstrap	0.0079	0.8866	5.0	Yes
Kfold	0.008	0.9119	10.0	Yes

When applying Lasso regression, the error terms do not improve, and do have less optimal values than Ridge regression. It also follows the same trend as for OLS and Ridge when calculating the error terms for the terrain data, as the R2-score gets even lower with $R2 = 0.6393$ and $MSE = 0.0452$.

Until now, the study of the error terms has only been performed with a fixed hyperparameter. The MSE as a function of the hyperparameter λ has therefore been studied further for Ridge and Lasso regression in the fifth polynomial degree. From the figure 5a, we can see that the R2 score remain quite similar to the OLS-regression, $R2 = 0.9269$, for $\lambda < 10^{-4}$. In the lasso regression on the other hand, it is observed that that the MSE value increases significantly when $\lambda > 10^{-6}$.



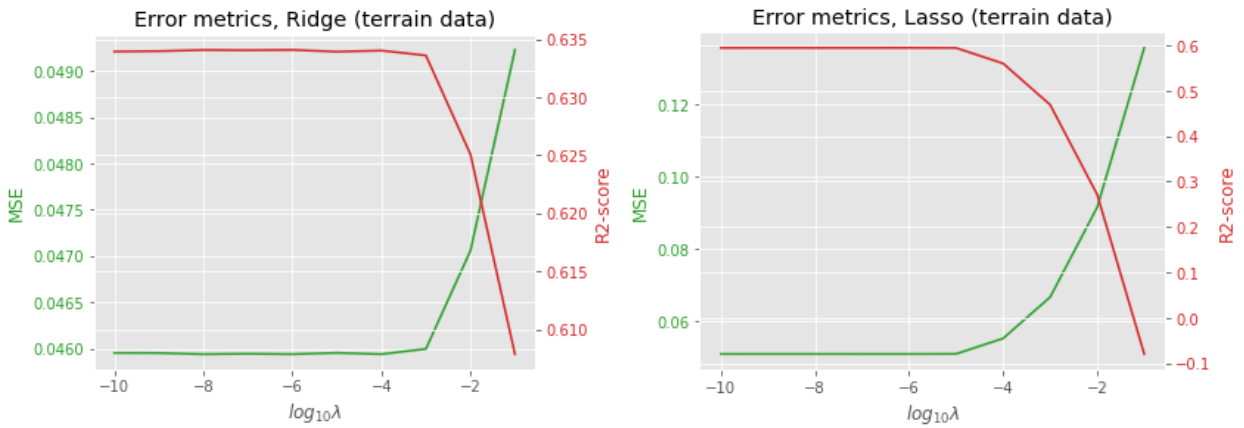
(a) Train vs test MSE

(b) Bias variance OLS

Figure 6: Error metrics of Franke's function using Bootstrap

As for the terrain data, the error metrics seems to remain quite stable and do mostly follow

the OLS-metrics. For Ridge regression, the error metrics starts to diverge when $\lambda > 10^{-3}$. Smaller hyperparameters are needed in order to remain the same error metrics as OLS when performing lasso regression, as the error metrics diverges when $\lambda > 10^{-5}$

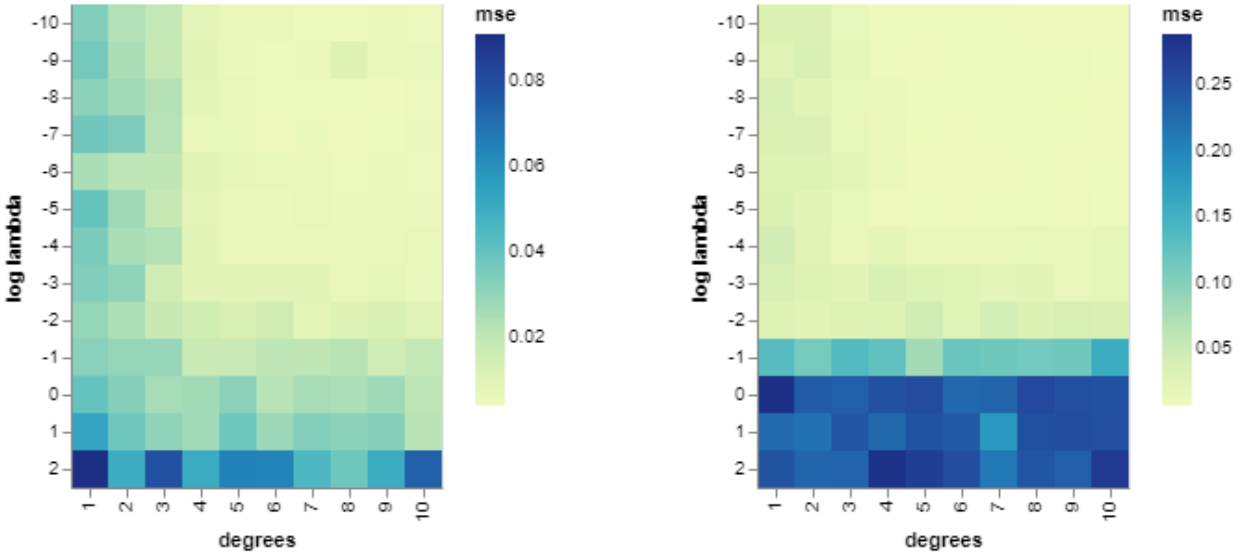


(a) MSE and R2 score, Ridge

(b) MSE and R2 score, Lasso

Figure 7: Error metrics of terrain data using Kfold cross-validation

For the purpose of studying the hyperparameter's further, a heat map was created in order to investigate the optimal hyperparameter values for Ridge and Lasso. The heat maps below shows that the MSE remains at a low value if the polynomial degree is 4 or higher, and if the hyperparameter $\lambda \leq 10^{-3}$ when applying Ridge regression. For lower order polynomials up to the 3rd degree polynomials, the MSE tends to change occasionally depending on the given value for the hyperparameter. On Lasso regression on the other hand, the MSE remains very low regardless of model complexity, but seems to drastically increase once $\lambda \geq 10^{-1}$ for all degrees.



(a) Rigde (b) Lasso

Figure 8: Heat maps of the Franke function between degrees, log lambda, and the MSE

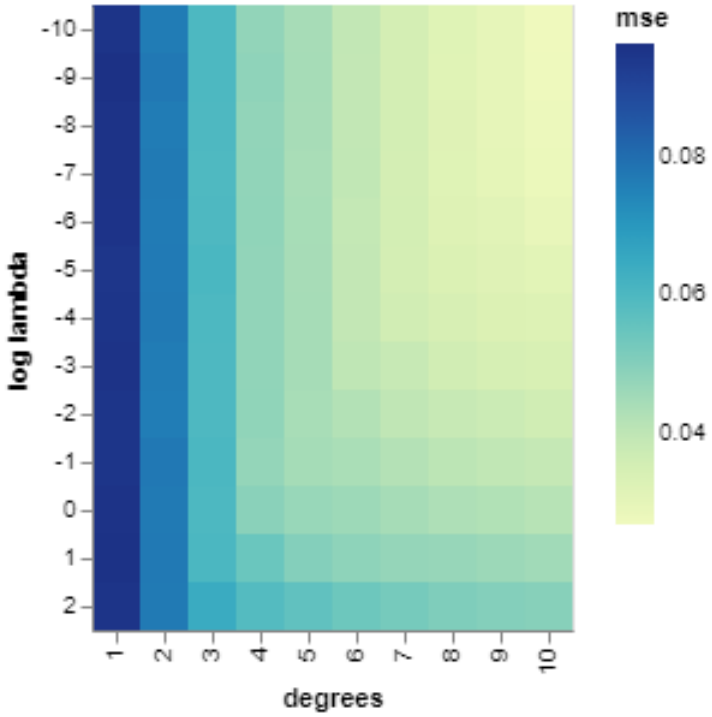


Figure 9: Heat map for terrain data, Ridge

The heat map of the terrain data shows a different pattern regarding the choice of hyper-parameters. Here, the MSE is uniformly high for the lower degree polynomials independent

from the choice of λ-values. For higher degree polynomials, the MSE improves gradually as the hyperparameters gets smaller. From figure 9, it almost appears that the optimal MSE is within the 10th degree polynomial with a hyperparameter value of $\lambda = 10^{-10}$. This indicates that the optimal MSE lies beyond the defined heat map.

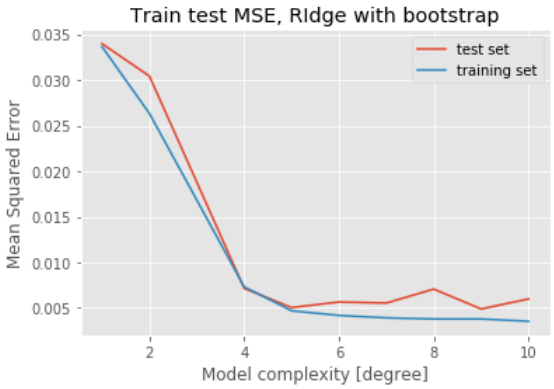
4.4 Bias-variance tradeoff

One of the purposes of plotting the bias and variance as a function model complexity (number of polynomial degrees) is to check if data is being overfitted. Based upon how the testing error seems to have a slightly increasing trend while the training error continues to decrease for all regression methods as shown in the figures below, this seems to be the case.

Figure 10: Train and test MSE of the Franke function as a function model complexity

(a) OLS

(b) Ridge



(c) Lasso

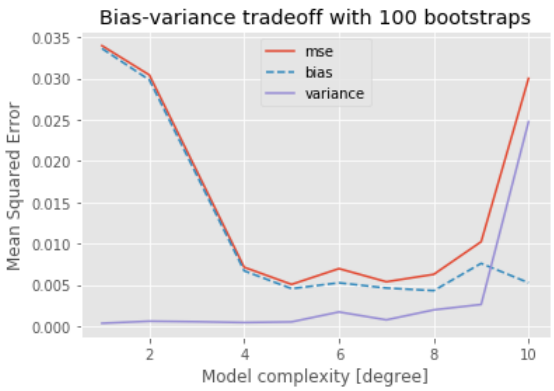
All of graphs are performed with 100 bootstraps, with a hyperparameter of $\lambda. = 10^{-5}$ for

Ridge and Lasso. The increase for test error are most notable for the OLS-regression where the test error curve forms an U-shape. The test error first decreases for lower degrees, hits a minimum, and then increases rapidly for the higher degree polynomials. It is also slightly notable for the Ridge regression, but the slope is somehow flatter, indicating that the presence of overfitting is not as strong compared to OLS. The lasso regression also seems to follow a similar trend to the Ridge regression. However, all the test MSE seems to reach the same minimum at a polynomial degree of 5.

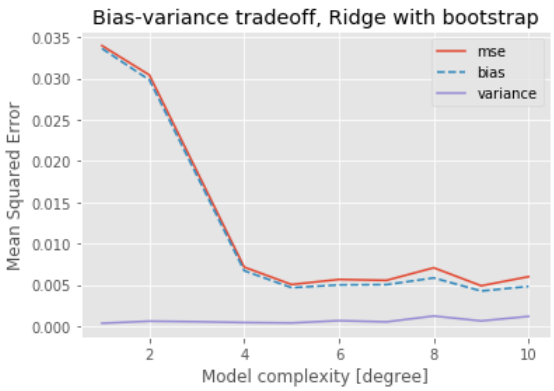
The separate components of MSE has been analyzed by using the bootstrap resampling method as discussed in section 2.8.1. The results of the bias-tradeoff analysis for the regression models are seen on figure 9 below.

Figure 11: Bias-variance tradeoff of the Franke function as a function model complexity

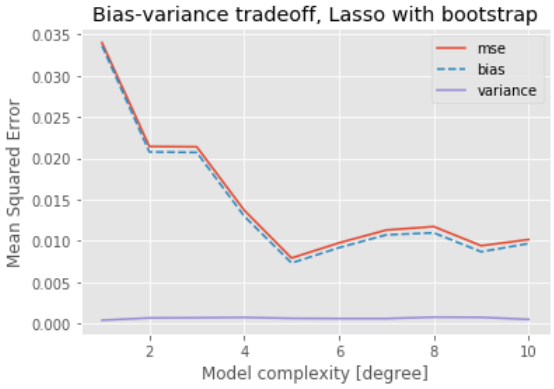
(a) OLS



(b) Ridge



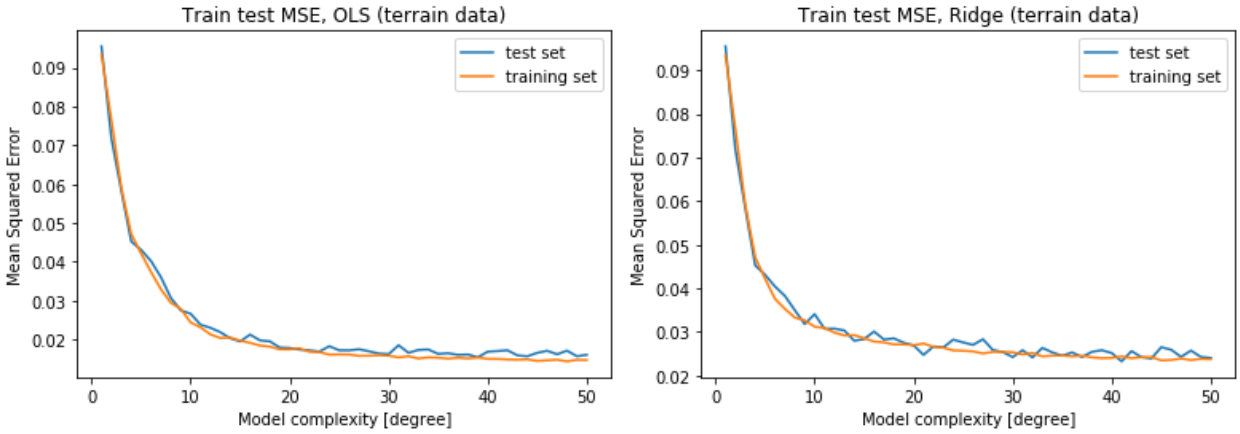
(c) Lasso



The biases are quite high for the lower complexities of all regression models, which means that the data are being underfitted. With increasing model complexity, the bias begins to decrease while the variance increases significantly for the OLS-model. The model in OLS on

the other hand starts to overfit at higher polynomial degrees, which is proved by the increase of variance followed by the sudden increase of the test MSE. The bias seems to slightly starts decreasing as well for the OLS. In the Ridge and Lasso regression, the bias tends to start increasing a bit for higher complexities, which is not expected. The variance of the Ridge and Lasso regression also seems to remain flat, which might indicate that it takes even higher model complexities before the variance starts to increase for Ridge and Lasso.

Analyzing the terrain data are deemed to be more demanding as the datasets contains a larger number of data points, leading to numerical limitations on complexity. Due to the limitations of computational resources, the model complexities have only been studied for OLS and Ridge only when studying model complexities higher than 20 degrees, and Sklearn's Kfold algorithm was used instead in order to reduce runtime as it is suspected that the use of if-statements in the implemented version of Kfold might have hampered the runtime. However, based upon the error metrics of the regression models, it seems like that the error decreases for higher model complexities, given the declining trend of the MSE. Another observation is that the variance is significantly low, and the bias remains high, which might indicate that the predicted models up to 10 degrees are too simple to give a good representation of the terrain, which was clearly seen from figure 3b.

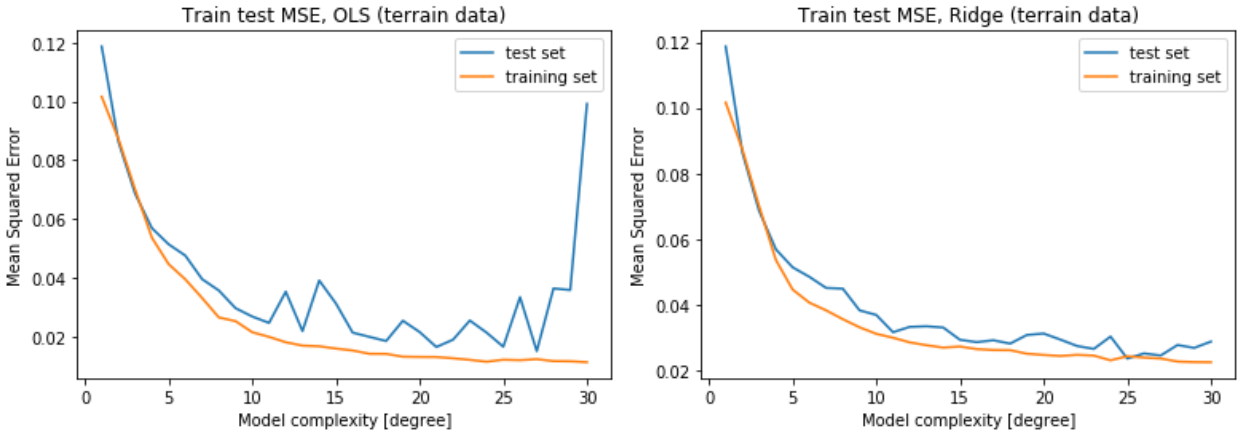


(d) MSE OLS

(e) MSE Ridge

Figure 12: Training and testing MSE for higher order polynomial up to 50 degrees

Furthermore, the plots of the training and testing MSE of OLS and Ridge looks identical to each other at this point when performing Kfold cross validation. The plots indicates no signs of overfitting even for very high model complexities, which could potentially mean that overfitting only will occur at a very large level of complexity that perhaps even might be impossible to numerically represent.



(a) MSE OLS

(b) Ridge OLS

Figure 13: Training and testing MSE with terrain image downsampled to 80×80

Since it was not possible to find signs of overfitting of terrain data for images with a finer resolution, the terrain data was made coarser by adjusting the pixel blocks to 80×80 This leads to the same expected overfit for higher complexities in OLS as shown in previous plots. The trend for OLS are similar to the bootstrap resampling of Franke's function in figure 8a. The Ridge regression on the other hand seems not to be too much affected by the change.

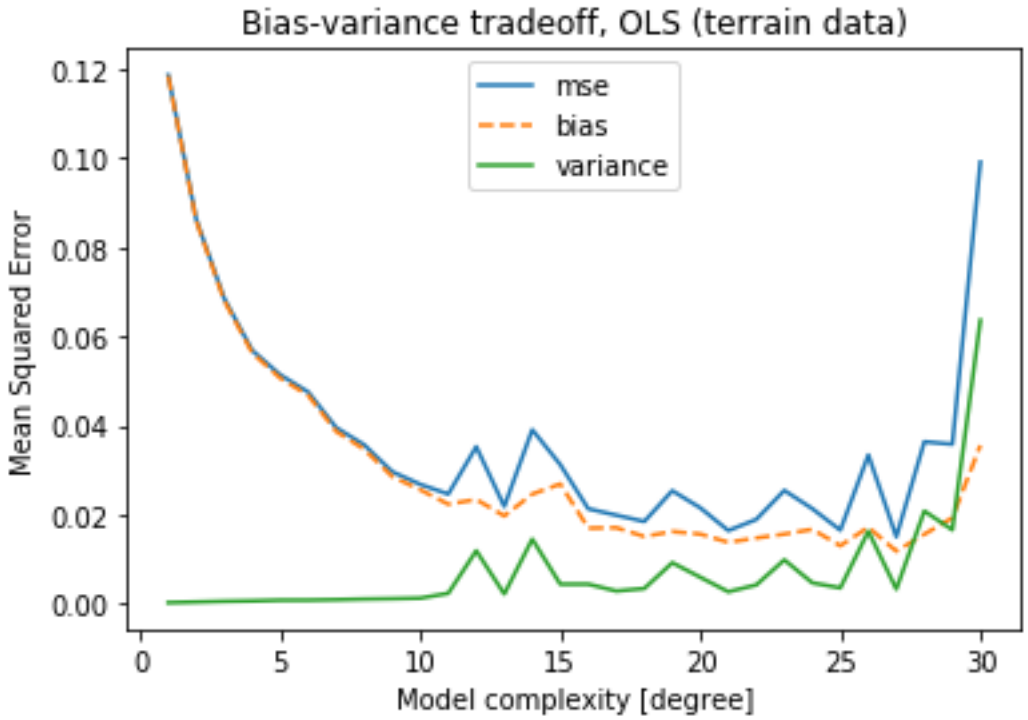
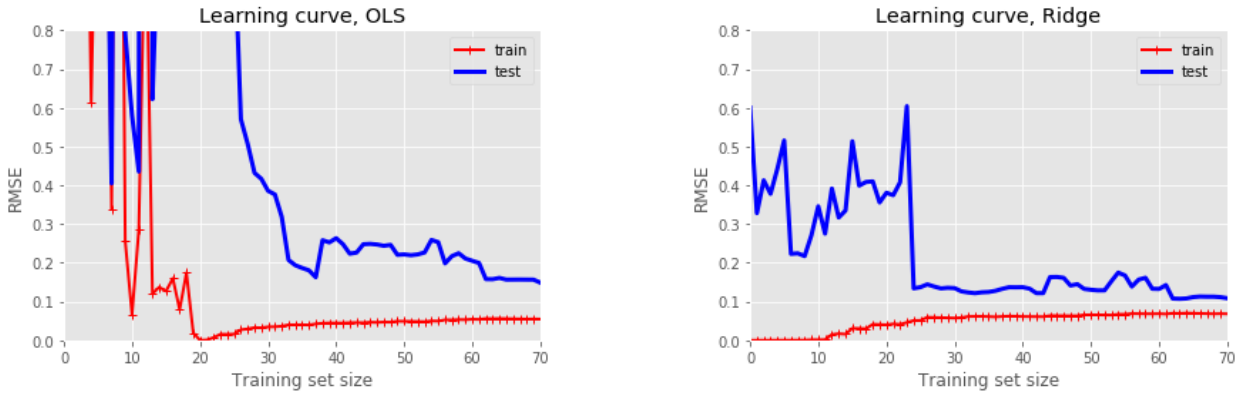


Figure 14: The bias variance tradeoff of terrain data with block size 80×80

By analyzing the components of the MSE generated from OLS regression further in figure 12, it is observed that the bias appears to have identically the same high values as the MSE in less complex models while gradually decreasing as the number of degrees of freedom increases. At the same time the variance stays relatively low and then starts to rapidly increase for more complex models. Unexpectedly, the bias also begins to gradually increase as well between degrees 20 and 30.

4.5 The learning curves



(a) OLS

(b) Ridge

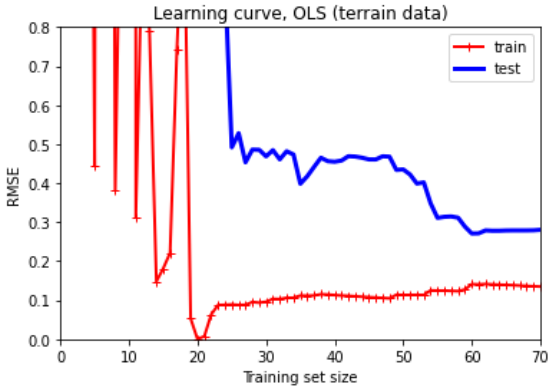
(c) Lasso

Figure 15: The resulting learning curves for the Franke function

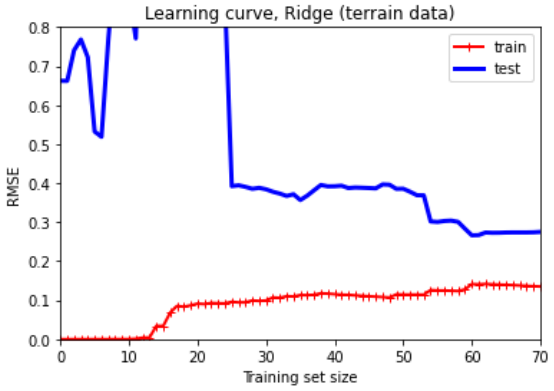
The resulting curves above shows the learning curves of each regression method for the Franke function. With few instances for the training set, the models are able to fit training data perfectly for Ridge and Lasso regression, hence the curves will start at zero. Once newer instances are added to training set, it becomes harder for the model to perfectly fit the training data, and thus the RMSE for the training sets begins to increase until it starts stabilize at some point, flattening the error curve. Once the training error curve flattens,

adding more data to fit does not improve or worsen the RMSE. The test error on the other hand remains high for all regression methods when the training set size is small, making it incapable to generalize properly. Once the model starts to learn by increasing the number of instances, the test error will decrease steadily, and also starts to flatten again at some point.

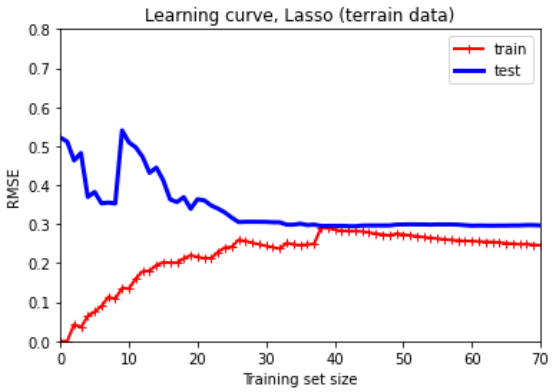
The usual learning curve pattern was however not the case for OLS, where the RMSE for the training set followed the same oscillating pattern along the test set in smaller traing set size. The reasoning behind this remain unexplained.



(a) OLS



(b) Ridge



(c) Lasso

Figure 16: The resulting learning curves for the terrain data

The same pattern can be observed from the terrain data with a block size of 30×30. However, it seems like the training and testing error do converge at a lesser extent for OLS and Ridge. An increase of the training sample size appears to stabilize the train error, but fails to make the test error to converge with the train error, indicating a high variance in the model, which in turn means data is being overfitted. This seems to be less problematic for the lasso regresion. Although Ridge regression tend to show some signs of overfitting, the test error might potentially converge with the train error eventually for higher number of instances. It could also be expected that it takes more instances before the train and test error converges

in OLS and Ridge regression if the terrain data is made coarser, i.e increasing the block size of the image.

5 Conclusion

The aim of this project was study the properties and behaviors of the regression models: OLS, Ridge, and Lasso, particularly by investigating the bias-variance tradeoff, and the model's tendency to overfit or underfit data. Common resampling techniques such as kFold cross validation and bootstrap were taken into used and applied in a generic dataset of the Franke function and a terrain map of Western Hokkaido, Japan. Bootstrap were applied in for both the Franke function and terrain data for all regression methods.

Based upon the results and by analyzing the test metrics, OLS proves to generate the lowest test error among the three methods for both the generic case in Franke function and for real terrain data. The OLS reaches an optimal R2 score at the fifth degree with $R2 = 0.955$ and $MSE = 0.004$ when applying Kfold cross validation.

Although an optimal polynomial degree for the Franke function was found, it is not quite possible to determine the same for the terrain data as the optimal polynomial degree could be arbitrarily high depending on the resolution of the digital elevation model. When studying the model complexity up to the 10th degree with resampling for the terrain data, the R2 score reached a maximum value of $R2 = 0.774$ and a mean squared error of $MSE = 0.0283$. The 10th degree was defined as the optimal value. This is however most likely not true, as the plotted model generated a much smoother terrain compared to the real terrain, which proves that a much higher degree is needed to properly fit the model.

The OLS-regression might perform better under the circumstances of the Franke function, but also tends to overfit data regardless of resampling method. This was also true for the coarser terrain data in higher order degrees as shown in figure 10a. Ridge and Lasso on the other hand do not show the same trends of overfitting the data compared to OLS. Ridge shows some weak signs of overfitting in the learning curves while Lasso barely had any increase of variance and remained flat for the most part as shown in figure 10c. Lasso regression does however have issues with runtime for larger sets of data. Being an iterative method with no analytical solutions for the β-coefficients made the calculations computationally expensive to solve. Although OLS might indicate the best fit for the data sets, it should be noted that a fine-tuned optimalization of the hyperparameter λ has not been discovered, as the general regression analysis for Ridge and Lasso had a fixed hyperparameter $\lambda = 10^{-5}$. Based on the heatmaps from figure 9, there are multiple hyperparameter values that could serve as a better candidate for fitting the data.

It is also noteworthy to mention that the tendency to overfit the terrain data depends on the resolution of the image. When downsampling the image to a finer resolution, 15×15 in

this case, none of the models showed signs of overfitting up to a model complexity of 50th degree polynomials. Resizing the blocks to make a much coarser image 80×80 do however show signs of overfitting already from the 30th-degree polynomial in the OLS-regression.

6 Future improvements

There were some issues with runtime regarding the usage of the own Kfold-cross validation method for the regression analyses of the terrain data. Furthermore, the implementation of the cross validation method was rather simple by only partitioning the data into 5 folds, 4 folds for training and 1 fold for testing, and calculate the average MSE once all possible distributions of folds are performed. Such methods were criticized by Varoquaux (2017) since the estimates of variance across cross validation folds underestimates errors on the prediction accuracy. A more suitable way of improving the current CV code immplemented in this report would be keeping the test set in a vault and create five validation set for finding the best models as a function of complexity and hyperparameters.

References

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edn, O'Reilly Media, Inc.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). The elements of statistical learning, *Aug, Springer* **1**.

Papachristoudis, G. (2019). The bias-variance tradeoff.
URL: *https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9*

Varoquaux, G. (2017). Cross-validation failure: Small sample sizes lead to large error bars, *NeuroImage* **180**.