Project 1 - Regression analysis and resampling methods

Espen Ulset Nordsveen Ghadi Al Hajj

October 8, 2020

Abstract

The Machine Learning field has seen great advancements in many scientific fields and many new algorithms are introduced each year. Linear Regression, with its three varieties, OLS, Ridge and Lasso, serves as the backbone of many advanced Machine Learning algorithms such as Logistic Regression, Neural Networks, etc. Therefore, studying this algorithm, with its simplicity compared to the other algorithms, is essential to build a robust foundation and understanding of the more advanced methods. In this project, we present an explanation of this method with applications on simple generated data and real terrain data, in order to compare the performance of the different varieties and present an analytical assessment of each. This was done with the help of re-sampling techniques, namely bootstrap and cross validation. We found that Ridge regression gave the best performance on the generated data, while OLS was superior on the terrain data. These results suggest that there is no single method that always performs the best in all cases, and that, indeed, each method would be more suitable in specific cases than the others.

1 Introduction

Linear regression is an important tool within the field of statistics and data analysis. This statistical method aims to determine the strength and character of the relationship between one dependent variable, also called the outcome, and one or more independent variables, also called the features. This is achieved by mapping this relationship through a mathematical formula that best approximates the data points within a given data set.

This tool is used in several fields of science spanning medicine, physics, finance, astronomy...

In this project, our goal was to analyse and compare the performance of various linear regression methods, namely the Ordinary Least Squares (OLS), Ridge regression and the Lasso regression. These methods were combined with resampling techniques like the bootstrap and cross validation techniques to get better estimates of the error, specifically on the test data. In addition, bootstrap was used to perform a bias-variance tradeoff analysis for all methods.

1.1 Linear Regression

As mentioned, linear regression attemps to find a relationship between two or more variables by fitting a linear equation to observed data. We denote the dependent variable, or outcome, as $\mathbf{y}$. If we have a set of independent variables, or features, $\mathbf{X}^T = (X_1, X_2, \ldots, X_p)$, and assuming a linear relationship between these, the y can be predicted as

$$\tilde{y} = \tilde{\beta}_0 + \sum_{j=1}^{p} X_j \tilde{\beta}_j \tag{1}$$

By including the constant variable 1 in X, and include $\tilde{\beta}_0$ in the vector of coefficients $\tilde{\beta}$, (1) can be written in vector form as

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{2}$$

where ϵ denotes the residual error of the linear model $\mathbf{X}\beta$ from the true response, and the β is the parameter vector containing the linear regression coefficients β_i. y and ϵ are defined as the following vectors

$$\mathbf{y} = [y_0, y_1, y_2, \ldots, y_{n-1}]^T \tag{3}$$

$$\beta = [\beta_0, \beta_1, \beta_2, \ldots, \beta_{n-1}]^T \tag{4}$$

$$\epsilon = [\epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_{n-1}]^T \tag{5}$$

and β are in fact the unknown parameters in the linear regression problems that needs to be calculated. The X matrix in (2) can be expressed as

$$\mathbf{X} = \begin{bmatrix} 1 & x_0^1 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1^1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2^1 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{n-1}^1 & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{bmatrix} \tag{6}$$

With the design matrix defined as (6), we can use this together with the unknown β to approximate y as

$$\tilde{\mathbf{y}} = \mathbf{X}\beta \tag{7}$$

We see that

$$\tilde{y} = \mathbf{X}\beta \tag{8}$$

$$= y - \epsilon \tag{9}$$

We therefore need to find β values such that the error ϵ is minimized, which indeed is the true objective of linear regression.

1.2 Ordinary Least Squares

The ordinary least square (OLS) method is defined by the following cost function [Hjort-Jensen]

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\{(\mathbf{y} - \tilde{y})^T(\mathbf{y} - \tilde{y}\} \tag{10}$$

In matrix-vector notation, this becomes

$$C(\beta) = \frac{1}{n}\{(\mathbf{y} - \mathbf{X}\beta^T\mathbf{y} - \mathbf{X}\beta)\} \tag{11}$$

By minimizing this cost function, this is in fact a estimation of the parameters β by minimizing the sum of the squared residuals. The minimization problem of the cost function (11) is solved by derivation of the function. The derivatives of (11) gives the parameterization of β for the OLS method as

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{12}$$

We see from (12) that finding an approximation for β involves a matrix inversion. In cases where $\mathbf{X}^T\mathbf{X}$ is singular, it is non-invertible, and the parameterization of β in (12) can not be calculated. In these cases, the *Singular Value Decomposition* (SVD) can be used. The SVD decompose any matrix $\mathbf{X}$ with dimension $P \times n$ into terms of a diagonal matrix and two orthogonal matrices, leading to the expression [Hastie et al.]

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{13}$$

Here $\mathbf{U}^{N\times p}$ and $\mathbf{V}^{p\times p}$ are orthogonal matrices, with the columns of $\mathbf{U}$ spanning the column space of $\mathbf{X}$, and the columns of $\mathbf{X}$ spanning the row space. $\mathbf{D}^{p\times p}$ is a diagonal matrix. In this study, SVD was used wherever the resulting matrices were non-invertible using the `np.pinv()` function from the `numpy` library.

1.3 Bias-Variance Tradeoff

For continuous predictions, we can assume that the true data is generated from a noisy model as [Hjort-Jensen]

$$y = f(x) + \epsilon \tag{14}$$

where ϵ is normally distributed with mean zero and standard deviation σ^2. We saw from the OLS method in section 1.2 that f can be approximated in terms of β and the design matrix $\mathbf{X}$ as $\tilde{\mathbf{y}} = \mathbf{X}\beta$. We then found the parameters β by minimizing the cost function

$$C(\mathbf{X}, \beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}[(\mathbf{y} - \widetilde{\mathbf{y}})^2] \tag{15}$$

This can be rewritten as

$$\begin{aligned}
C(\mathbf{X},\beta) &= \mathbb{E}[(\mathbf{y}-\widetilde{\mathbf{y}})^2] \\
&= \mathbb{E}[(y-\mathbb{E}[\tilde{y}]+\mathbb{E}[\tilde{y}]-\tilde{y})^2] \\
&= \mathbb{E}[(y-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})^2]+2\,\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})(y-\mathbb{E}[\tilde{y}])] \\
&= \mathbb{E}[(y-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})^2] \\
&= \mathbb{E}[(f+\epsilon-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2] \\
&= \mathbb{E}[f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[\epsilon^2]+2\,\mathbb{E}[(\epsilon)(f-\mathbb{E}[\tilde{y}])^2)]+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2] \\
&= \mathbb{E}[f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[\epsilon^2]+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2] \\
&= \mathbb{E}[f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[\epsilon^2]+\mathbb{E}[(\tilde{y}-\mathbb{E}[\tilde{y}])^2] \\
&= Bias^2+\sigma_\epsilon^2+Variance \\
&= \frac{1}{n}\sum(f_i-\mathbb{E}[\tilde{\mathbf{y}}])^2+\frac{1}{n}\sum(\tilde{\mathbf{y}}-\mathbb{E}[\tilde{\mathbf{y}}])^2+\sigma^2
\end{aligned}$$

We see that the first part is the difference between the expected value of the model prediction and the true value of y, and is called the *bias*. The value is used for evaluation of the predicted model. A bias of zero is called unbiased, and will thus not contain any systematic or random errors, which is fairly unlikely. The second part is the squared deviation of y from its mean, which is called the variance. It basically describes the spread of the prediction.

1.4 Ridge Regression

Ridge regression is a method for shrinking the regression coefficients by imposing a penalty on their size. The Ridge coefficients minimize a penalized residual sum of squares. This circumvent the problem with colinearity we saw for the OLS method, where $\mathbf{X}^T\mathbf{X}$ might be singular. This is especially true for higher dimensions of the design matrix. By adding a penalty factor λ to the diagonal of $\mathbf{X}^T\mathbf{X}$, the term is always invertible. The cost function for ridge regression can be expressed as

$$C(\mathbf{X},\beta)=\frac{1}{n}\{(\mathbf{y}-\mathbf{X}\beta^T\mathbf{y}-\mathbf{X}\beta)\}+\lambda\beta^T\beta \tag{16}$$

and the parameterization for the ridge method thus become

$$\beta=(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{17}$$

We see that a diagonal matrix including the penalty factor λ is added to $\mathbf{X}^T\mathbf{X}$.

1.5 Lasso Regression

As for Ridge regression, Lasso is also a shrinkage method. Lasso stands for "Least Absolute Shrinkage and Selection Operator", and like Ridge, it fits the linear regression model by minimizing the sum of the squares augmented with

a penalty factor. However, in Lasso regression, the penalty function becomes non-linear. In contrast to the Ridge regression, it is in fact the absolute values of the regression parameters multiplied by the penalty factor λ [Wieringen]. There is no analytical expression for finding β, but is in fact done iteratively.

1.6 The Bootstrap

The boostrap resampling method is a simple Monte Carlo technique to approximate the sampling distribution [Murphy]. The method randomly draw B datasets with replacement from the training data, where each sample is the same size as the original training set. The model is then refit to the B bootstrap datasets [**Hastie et al.**]. In the context of this project, we use this method to calculate MSE, bias and variance of the model.

1.7 *k*-fold Cross Validation

The k-fold Cross Validation technique is a resampling procedure used to evaluate the performance of statistical models on a limited data sample. The method generally results in a less biased or less optimistic estimate of the model compared to other methods, such as the train/test split. The k refers to the equally number of groups the given sample shall be split into. One of these groups are set aside as the validation group. The rest of the groups, i.e. $K - 1$, are used to fit the model, and calculate the prediction error of the fitted model when predicting the kth part of the data. A normal value for k is $K = 5$, as this gives a lower variance. It might, however, give problems with the bias if the training sets becomes too small.

The general procedure for the k-fold Cross Validation is as follows

1. Split the dataset into k groups
2. For each unique group
   (a) Take the group as a hold out or test data set
   (b) Take the remaining groups as a training data set
   (c) Fit a model on the training set and evaluate it on the test set
   (d) Retain the evaluation score and discard the model
3. Summarize the skill of the model using the sample of model evaluation scores

The value of k is normally set to 5 or 10.

1.8 Model analysis

To evaluate how well the predicted model fits the real data, we need to define some measures to be evaluated

1.8.1 Mean Squared Error

The mean squared error (MSE) of an estimator is the averaged squared difference between the estimated values and the actual value. That is, the difference between the predicted data points $\tilde{\mathbf{y}}$ and their actual value $\mathbf{y}$, and is defined by

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \tilde{y}_i)^2 \tag{18}$$

MSE values close to zero thus indicates a pretty good estimate for the model.

1.8.2 $R2$-score

The R2 value, or the *coefficient of determination*, measures how well future values are likely to be predicted by the model. It is a relationship between the variance of the predicted model and the total variance of the input data, and is defined by

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{\mathbf{y}}_i)^2} \tag{19}$$

where $\overline{y}$ is defined as

$$\overline{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{20}$$

The R^2 score might be positive or negative. A score of 1 is the best possible score, and indicates that the predictions perfectly fits the data. A constant model that always predicts the expected value of $\tilde{y}$ without caring for the input data, get a R^2 score of 0.

1.9 Confidence interval

The confidence interval gives an estimate of how well β is predicted. The 95% confidence interval for β_j can be defined as [Hastie et al.]

$$c_{\beta_j} = \beta_j \pm 1.645\sigma \tag{21}$$

It gives an confidence level that the true parameter is in the proposed range.

2 Data

2.1 Franke's function

The Franke's function will be used as a basis for our discussion of the various regression and resampling methods in this context. This is a function widely used as a test function in interpolation problems, and is defined by

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

3 Method

The Franke function is a multivariable function, depending on both x and y, and hence is a 2-variable polynomial. The design matrix in (6) thus needs to be adjusted for this multivariable case. The design matrix thus becomes

$$\mathbf{X} = \begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 & \dots & x_0^N & y_0^N \\ 1 & x_0 & y_1 & x_0y_1 & \dots & x_0^N & y_1^N \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 1 & x_1 & y_0 & x_1y_0 & \dots & x_1^N & y_0^N \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 1 & x_{n-1} & y_{m-1} & x_{n-1}y_{m-1} & \dots & x_{n-1}^N & y_{m-1}^N \end{bmatrix}$$

where N is the polynomial degree.

4 Results

4.1 Ordinary least squares on the Franke Function

In this part, we performed the plain OLS regression on the data that is generated using the Franke function. Before that, we split the data into training set, on which the model is trained, and a testing sets, on which the model's performance is evaluated for the goodness of the fit. The ratio of training to testing sets 70/30. The same train/test split was done in all the coming parts to ensure a fair and reproducible comparison between the different methods. This was achieved by using the same random seed in all the cells.

For performing the OLS, a polynomial of degree five was used to generate the features of the model, and then these features were scaled according to the training data statistics, in order to force the data into a distribution with zero mean and unit variance. This is useful because it ensures that each feature gets an equal chance of being represented in the polynomial, thus preventing features with large values to dominantly control the shape of the polynomial.

The metrics were then applied to both sets and the R2 coefficient and the Mean Square Error of each set were calculated. In addition, we generated a plot of the confidence intervals of the tunable parameters, shown in figure 2. The Mean Square error was calculated by (18) and the R2 coefficient was calculated by (19). In our demonstration, we noticed that if the design matrix is created from data points within the range of [0,1], then the resulting matrix would be

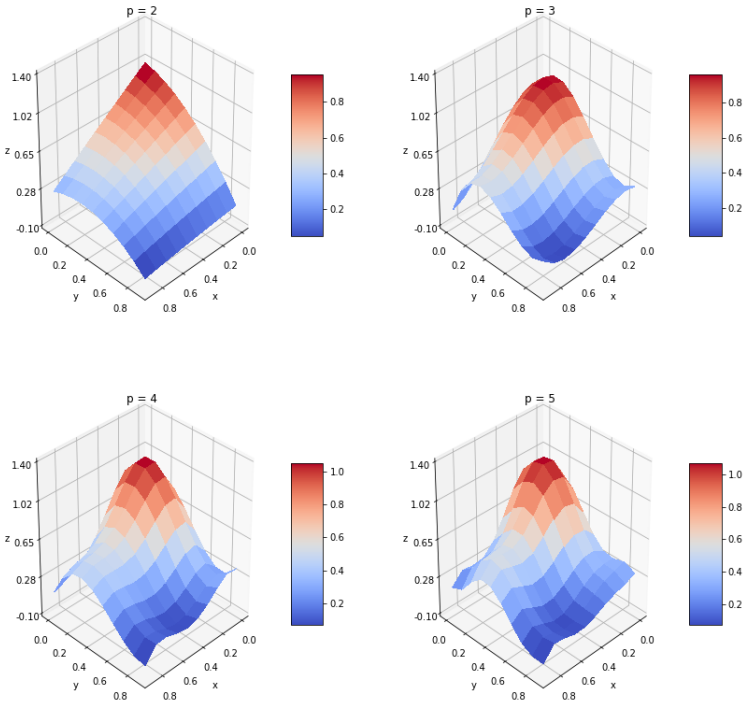


Figure 1: OLS on the Franke function, with increasing polynomial degree

close to singularity, and the program would thus raise an error whenever the inverse is calculated, particularly when calculating or its variance. This, in turn, caused the problem of getting negative values for the variance of some values. Using pinv, however, instead of inv, solved both problems when calculating the inverses.

It is also worth mentioning that when the range of values that the raw features can take is increased to, say, [0,5], instead of [0,1], the problem of singular matrices immediately disappears even when using inv and the determinant of the matrix becomes greater than zero.

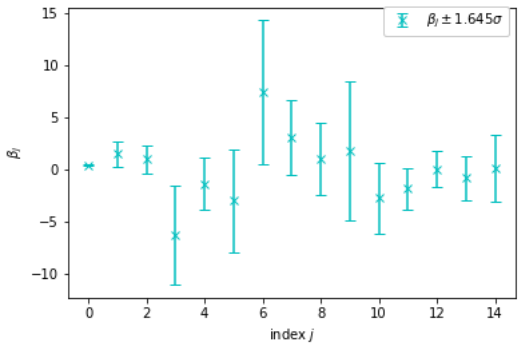


Figure 2: Confidence intervall of the tunable paramteres

4.2 Bias-variance trade-off and resampling techniques

When evaluating the model's performance on the test data that the model has not seen before, we want to measure the difference/error between the model's predictions and the true values. This error can be split into three terms.

The bias term is a quantity that describes how much the data is shifted from the mean value of the model. A high bias indicates that the model is not able to represent the patterns in the underlying data.

Variance, on the other hand, represents the variance of the predicted data from its corresponding mean, and it describes the sensitivity of the model to the noise in the data. Thus, high variance indicates that the model has not only learned the underlying model, but went further to even capture the noise in the data.

A model that is simple and does not capture the underlying trends in the data is called an underfitting model, and has high bias and low variance. On the other hand, a model that is too complex will likely result in overfitting, in which case it would have low bias and high variance.

Last is the irreducible error term. This represents the error that is not related to the complexity of the model and that cannot be reduced no matter how good and tuned the model is, with a given set of predictors. This error term is given by the variance of the noise in the data, and can be reduced by recognizing more independent predictors that are also related to the dependent variable.

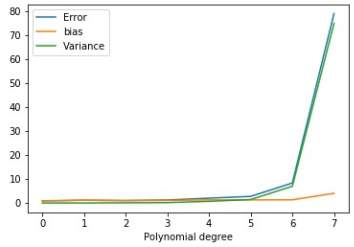
Figure **??** (add figure for bias variance tradeoff) shows the trends in the bias and the variance as a function of the model's complexity.

As expected, at low degrees of complexity, the model suffers from underfitting as a low order polynomial cannot reasonably represent a complex function with exponential and quadratic terms. This is shown by the high bias for polynomials of low order.

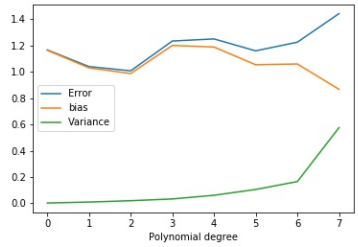
As the complexity increases, the bias starts decreasing indicating that the

model is no longer deficient of approximating the overall shape and trends in the data. At a certain polynomial degree (5), the model reaches a balance between bias and variance, where the overall error is the lowest. Such a model is a model that has learned the trends in the data but has not yet entered the stage of memorizing the training data and adapting itself to the noise/idiosyncrasies in the training data.

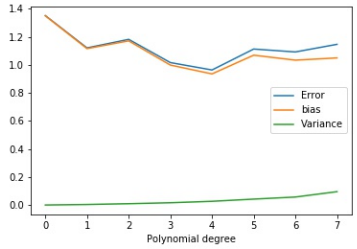
When the polynomial degree exceeds 5, the variance starts to increase in value indicating that the model has now entered into the realm of over-fitting. This variance will keep increasing as the complexity increases until the model has learned all the noise in the training data. This additional learning negatively impacts the model's performance on new data and inhibits its ability to generalize to the domain of interest, which is the ultimate goal of any machine learning algorithm.



(a) legend



(b) legend



(c) legend

Figure 3: Plots of the relationships between error, bias and variance with bumber of bootstraps = 200, using the bias-variance tradeoff

4.3 Cross-validation as resampling techniques

In this part, we performed a k-fold cross validation analysis to estimate the value of the Mean Square Error of our model of a seventh degree polynomial. We estimated this quantity using 5 and 10 folds.

The cross validation analysis gives estimates of 0.0033 and 0.0021, for 5- and 10-folds, respectively, which are both relatively lower than the estimate of this quantity from bootstrap at the same model complexity, 0.1019. This illustrates the fact that cross validation gives an over optimistic estimate for this error.

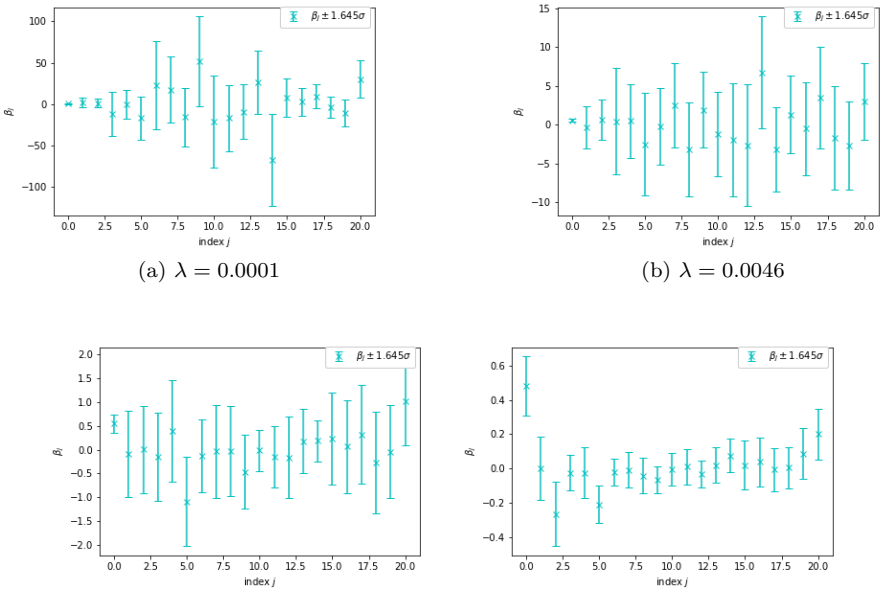
It is also important to note that the bootstrap approach is closer to the real scenario because after each bootstrapping iteration, the model is tested on the same test set, whereas in cross validation, each time the test set is changed.

4.4 Ridge regression on the Franke function with resampling

In this part, we used the same data from a Franke function to study the effect of L2 regularization, i.e. Ridge regression, on the output model and its performance.

For the hyper-parameter λ, we used four different values on a log-scale from -4 to 1 to create four different models. Specifically, these corresponds to the values 1.0 e-04, 4.6 e-03, 2.2 e-01, and 1.0.

From figure 4, it can be seen that as the value of λ increases the parameters beta and their confidence intervals become more centered around zero. This is because a higher value of places a higher penalty on high values of , thereby preventing it from taking large values, which can lead to over-fitting. In the



(a) $\lambda = 0.0001$ (b) $\lambda = 0.0046$

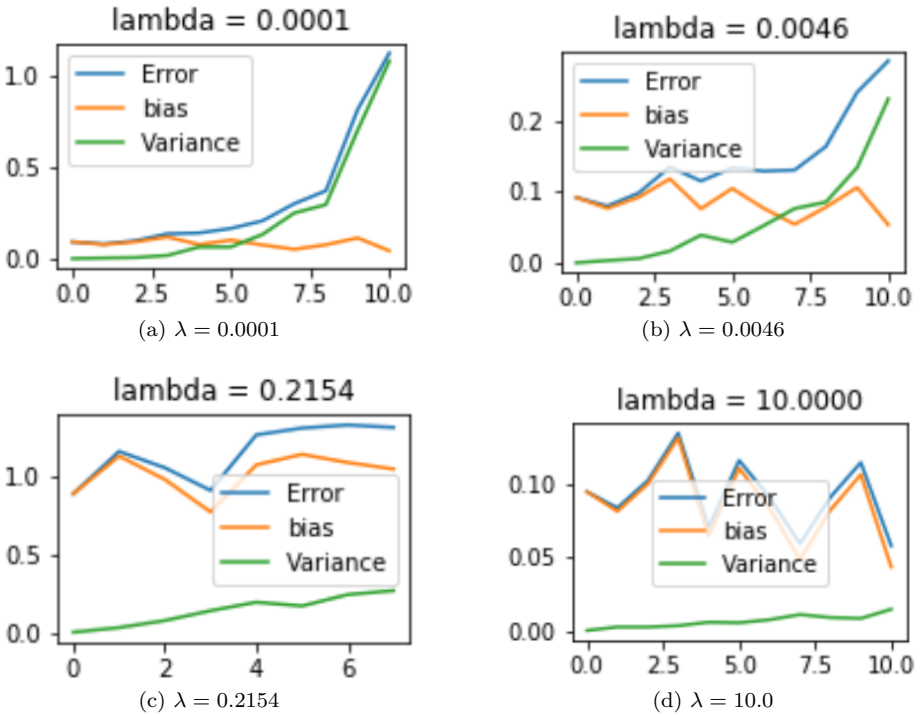
(c) $\lambda = 0.2154$ (d) $\lambda = 10.0$

Figure 4: Plots of the confidence intervall for the ridge regression on the Franke function

same analysis, comparing the values of the Mean Squared Error across the used values of shows that the error on the test set decreases with increasing values of λ whereas the error of the training set increases. This is because, as discussed before, L2 regularization fights over-fitting. Therefore, for higher values of , less emphasis is put on reproducing the training data set with its idiosyncrasies and more importance is placed on learning the general patterns in the training data. This, in turn, improves the model's ability to generalize to data points that it

has not seen before during training.

Next, we applied the bootstrapping method to ridge regression to compare it with the results from OLS regression, and study its behavior in the bias-variance trade-off. See figure 5. The same behavior of decreasing bias and increasing variance is observed, though, the bias's behavior is more evident at low values of noise variance. However, it it also important to note that as the value of λ increases it takes the variance additional degrees to emerge and manifest itself. This is, again, expected because regularization fights over-fitting, which a model exhibits when the variance is high. Moving into cross-validation, the



(a) $\lambda = 0.0001$ (b) $\lambda = 0.0046$

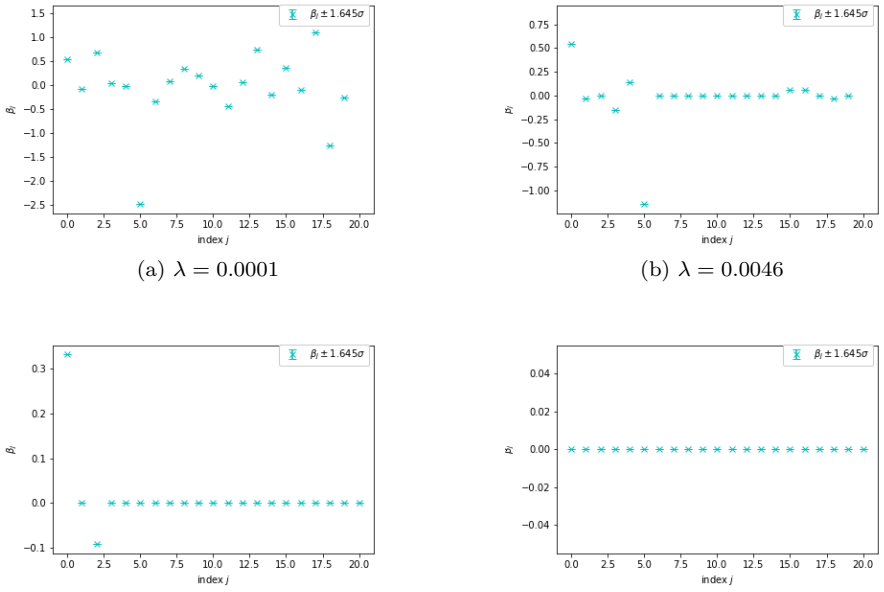
(c) $\lambda = 0.2154$ (d) $\lambda = 10.0$

Figure 5: Plots of the relationships between error, bias and variance using Ridge bootstrapping

ridge regression model gives a lower estimate of the MSE than OLS at all values of λ and both number of folds, except at $\lambda = 10^{-1}$ with five folds. Moreover, similar to the case with OLS, again, cross-validation gives a lower estimate of the error compared to bootstrap.

4.5 Lasso regression

Complementing our study of OLS and Ridge regression, here, we present a similar study on the Franke function data to study the effect of L1 regularization, i.e. Lasso regression, on the resulting model and its performance, see figure 6. Here, the values of λ are the same as those used in Ridge regression. Similar



(a) $\lambda = 0.0001$

(b) $\lambda = 0.0046$

(c) $\lambda = 0.2154$

(d) $\lambda = 10.0$

Figure 6: Plots of the confidence intervall for the lasso regression on the Franke function

to Ridge regression, as the value of λ increases the parameters beta become more centered around zero, for the same reason of being penalized for having large values. However, the true difference between Lasso and Ridge is that instead of just centering the values of beta around zero, thereby, preventing them from attaining large values, Lasso takes a step further and actually set some parameters to exactly zero. This behavior is more evident at high values of λ. In this way, Lasso can be seen as a shrinkage and feature selection method. Here, it is important to note that the confidence intervals were not calculated as there is no analytical equation for this quantity. Instead, only the values of the obtained β's are displayed. Comparing the models built with different values of λ, we find that Mean Squared Error on the test set decreases with increasing values of λ, whereas the error of the training set increases. The same reasoning follows here of how L1 regularization fights over-fitting by, thereby finding a balance between reproducing the training data and generalizing to data that it

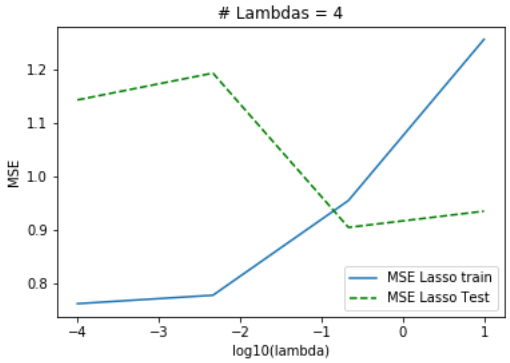


Figure 7: MSE for train and test set for Lasso

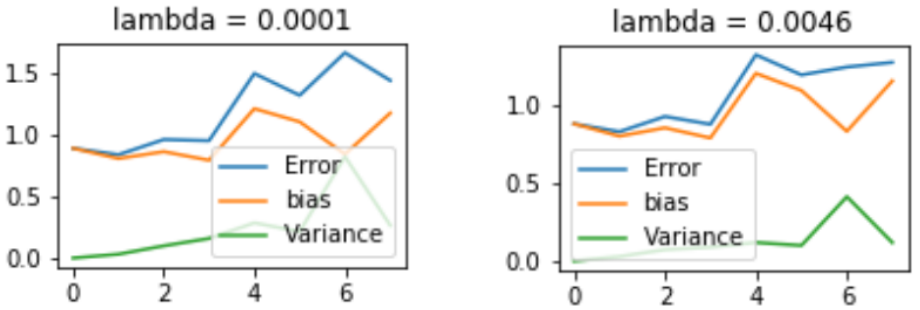
has not been trained on.

For the bias-variance trade-off, we observed a similar behavior of decreasing bias and increasing variance to that observed in Ridge. Again, it takes the variance higher complexity values to become evident.

Moreover, in figure 8 the cross-validation analysis shows that the Lasso regression model, again, gives a lower estimate of the MSE than OLS at all values of λ and both number of folds. Moreover, similar to the case with OLS and Ridge, again, cross-validation gives a lower estimate of the error compared to bootstrap, thereby giving more optimistic results.

Finally, given all the previous analyses, and based on the results from the bootstrap resampling method since it studies the behaviour of the regression method at different polynomial degrees, it can be seen the the polynomial degree 3 gives the best results among all the used methods. Moreover, Ridge regression gives the least MSE value of 0.8269 at that polynomial degree with $\lambda = 10$. Therefore this is selected as the best model.

For such an input surface, this result is reasonable since Ridge finds a perfect spot between the other two methods. Indeed, it adapts to data trends, similar to what OLS does, but at the same time, it takes the generalization of the model into consideration by penalizing large values of β, therby not learning the noise in the training data, though in a less extreme fassion than Lasso.



(a) $\lambda = 0.0001$ (b) $\lambda = 0.0046$

(c) $\lambda = 0.2154$ (d) $\lambda = 10.0$

Figure 8: Plots of the relationships between error, bias and variance using Lasso bootstrapping

4.6 Introducing real terrain data

Looking beyond the Franke functions from now, and looking into real data, the previous methods will now be tested on terrain data. We use digital terrain data from Norway, see figure 9 for a visual presentation.

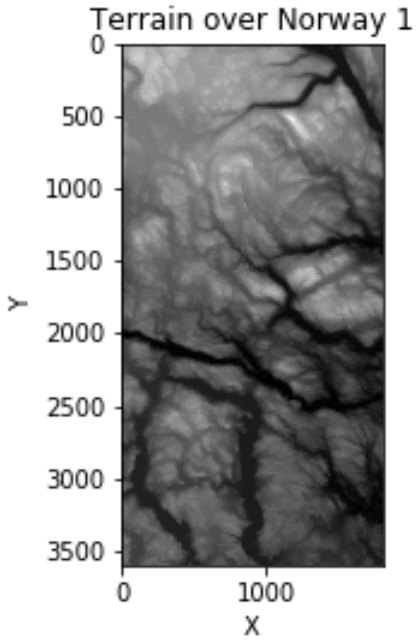


Figure 9: 2D plot of the terrain data

4.7 Regression techniques on terrain data

In this part, we applied all of the three discussed methods on the terrain data. In order to get a general sense of each method's approach in reproducing the terrain data, we used all of the cropped image to train a model of each type. All of the data was used because splitting the data and plotting the results of the training data would not be possible as the `sklearn.model_selection.train_test_split` function does not return the indices of the training data to be used in the plots.

The cropped region from figure 9 is shown in figure 10, entitled as "True Function". The learnt surfaces of all three methods trained on that region are shown corresponsingly in the same figure. For reasons to be discussed later in this part, we chose the degree 25 as our polynomial degree in building the design matrix. In order to select the best model, we performed the 10-fold cross validation analysis on the terrain data using the three models at different model complexities, and different λ values for regularization in Ridge and Lasso regression. The results are shown in table 1), 2 and 3.

Degree	MSE
5	487.4188
10	130.5130
25	62.3598
25	57.0601

Table 1: Mean MSE scores for OLS on the terrain data with increasing polynomial degree

	p = 5	p = 10	p = 25	p = 50
$\lambda = 10^{-4}$	487.4197	221.9297	138.3173	122.1696
$\lambda = 4.64^{-3}$	492.1422	261.9113	190.5044	175.4842
$\lambda = 2.15^{-1}$	564.7006	375.4598	276.1327	259.0530
$\lambda = 10$	640.9542	516.4857	427.2646	400.7308

Table 2: Mean MSE scores for Ridge on the terrain data with increasing polynomial degree and λ

	p = 5	p = 10	p = 25	p = 50
$\lambda = 10^{-4}$	617.2067	508.5664	432.8633	413.3999
$\lambda = 4.64^{-3}$	617.6259	508.2350	430.0946	412.0222
$\lambda = 2.15^{-1}$	717.3227	685.6355	620.3984	621.7994
$\lambda = 10$	4259.9043	4350.5270	4352.4810	4347.9143

Table 3: Mean MSE scores for Lasso on the terrain data with increasing polynomial degree and λ

This simulation shows that, for sensible reasons, low degree polynomials fail

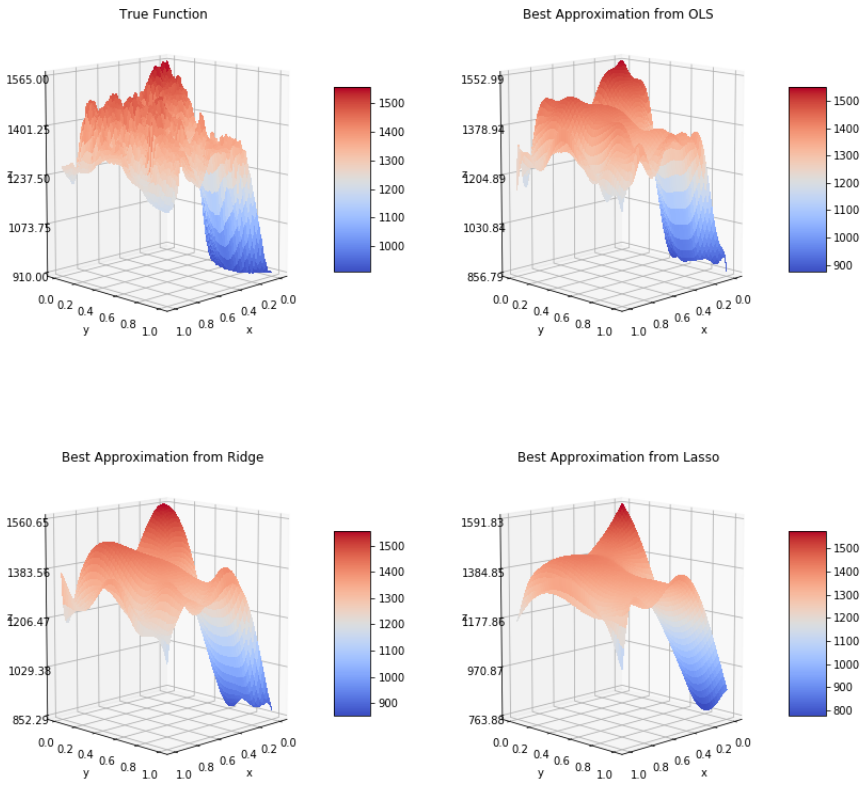


Figure 10: Prediction for terrain data

at learning, and thus reproducing, such a terrain with a large number of bends and spikes. Moreover, at all of the used polynomial degrees, OLS regression gives the best results among the used methods, as indicated by the low MSE values. The reason for that is, as already discussed, both Ridge and Lasso put penalities on large values of β, thereby smoothing the resulting curve. This is clearly visible in figure 10. Therefor, since the data is very noisy, learning a smooth function will not yield the best results, as such a function does not adapt to this noise.

Before selecting OLS as our final model, we performed simple search over higher polynomial degrees using OLS only, as it has shown the most promising results. and in order to avoid lengthy computation times required by the other methods at such high degrees. The table (add another table) shows that as we increase the degree from 25 to 55 in steps of 10, we observe a minimal decrease in the MSE, with a comparably higher increase in computation time. For the reasons discussed above, we chose OLS regression at degree 25 as our final model for this study.

References

[1] Trevor Hastie, Robert Tibshirani and Jerome Friedman *The Elements of Statistical Learning* Springer, Stanford, California, 2020

[2] Kevin P. Murphy *Machine Learning - A Probabilistic Perspective* The MIT Press, Cambridge, Massachusetts, 2012

[3] Morten Hjort-Jensen: Linear Regression and Review of Statistical Analysis and Probability Theory
`https://compphysics.github.io/MachineLearning/doc/pub/week35/html/week35.html` 2020

[4] Wessel N. van Wieringen: Lecture notes on ridge regression
`https://arxiv.org/pdf/1509.09169.pdf` 2020