Regression analysis and resampling methods

Søren B. Tvingsholm[1]

[1] *University of Oslo, Department of Physics, Norway*

Repository: https://github.com/SorenBTV/maskin/tree/main/proj1
Date: 07/10/2024

Abstract—This report investigates the application of linear regression techniques, including ordinary least squares (OLS), ridge regression and lasso regression. These techniques have been applied to synthetic data created by using the Franke function, and to real geological terrain data. The study is focused on evaluating the bias-variance tradeoff and improving the predictive accuracy of our regression models through resampling techniques such as Bootstrap and Cross-Validation. The analysis infers that the choice of regression method should depend on the dataset and the application of resampling techniques. OLS and ridge regression show distinct strengths, while lasso regression was found to require more computational power. The study highlights the importance of selecting the appropriate regression method based on dataset, with resampling techniques offering potential improvements in predictive accuracy.

I. Introduction

Linear regression are machine learning algorithms, which are capable of enhancing a programs performance through experience [1]. The following report is a result of studying various regression methods such as ordinary least squares (OLS), ridge regression and lasso regression when applied to our own data set given by the Franke function as well as on a real geological data set. The three different regression methods were evaluated using bias-variance tradeoff, and attempted improved by using Bootstrap and Cross-Validation as resampling techniques. In other words, the objective of this study is to explore and discuss which regression model and resampling techniques that are most optimal for the different data sets.
Primarily, the focus is to produce linear regression algorithms for the Franke function with some additional noise. Following will be adding and comparing resampling methods to the algorithms, considering our models functionality with bias-variance tradeoff. Lastly, the established programs were used to perform linear regression on the real geological data set. This is to be presented with relevant methods, results, discussion and conclusion respectively.

II. Method

Initialising the Franke function

To research the different regression methods, a test function needs to be used. In this case we set up the Franke function which is a sum of four exponentials:

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

Data for x and y consists of n amount of random numbers in the interval $[0,1]$, which are then set up in a $(n \times n)$ meshgrid. x and y are then processed in the Franke function to provide a data set z. To make the data set more realistic, additional stochastic noise is added $\varepsilon \sim N(0,\sigma^2)$ for a chosen standard deviation σ. The data provided by the Franke function and noise is then given by:

$$z = f(x,y) + \varepsilon.$$

Ordinary least squares (OLS)

OLS is a regression method that provides approximations for $z = f(x,y) + \varepsilon$ by minimizing $(z-\tilde{z})^2$, with $\tilde{z}$ being the predicted next function value, to give us the matrix equation

$$\tilde{z} = X\beta.$$

Here, X is the design matrix with dimensions $(n \times p)$ where n is the number of samples and p is the number of features. β is a vector of coefficients where each β_i determines how each feature affects our prediction. In our case we need to have a design matrix that follow both **x** and **y** as dependant

polynomials. Our design matrix takes the following shape:

$$X = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 & \dots & x_0^N & y_0^N \\ 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & \dots & x_1^N & y_1^N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & y_n & x_n^2 & x_ny_n & y_n^2 & \dots & x_n^N & y_n^N \end{bmatrix}.$$

Since the matrix is symmetric, our features p are equal to our number of samples n, making a $(n \times n)$ matrix.

To study the previously stated goal of having as small a discrepancy between the observed value and the predicted value, OLS evaluates the mean squared error (MSE). MSE is given by the equation

$$MSE(z,\tilde{z}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2.$$

For the MSE to be as small as possible we need to find the optimal β values. In order to find the optimal β value we need to use the cost function

$$C(\beta) = \frac{1}{n}\left\{(y - X\beta)^T(y - X\beta)\right\}.$$

Finding the derivative of the cost function and solving it for zero, gives us an equation for the optimal β values for OLS

$$\beta = (X^TX)^{-1}X^Tz.$$

Using OLS it is possible to find the expectation value(bias) and variance of our data. Our model will be evaluated by applying a bias-variance trade-off analysis. The expectation value $\mathbb{E}$ and variance for z_i and $\hat{\beta}$ are as follows

$$\begin{aligned} &\mathbb{E}(z_i) = X_{i,*}\beta, \\ &Var(z_i) = \sigma^2, \\ &\mathbb{E}(\hat{\beta}) = \beta, \\ &Var(\hat{\beta}) = \sigma^2(X^TX)^{-1}. \end{aligned}$$

For a detailed guide on how to derive the equations for expectation value and variance, see Appendix A.

Ridge regression

Ridge regression is a regression method similar to OLS, but with the addition of a regularization parameter λ in the cost function

$$C(X,\beta) = \left\{(y - X\beta)^T(y - X\beta)\right\} + \lambda\beta^T\beta.$$

Finding the derivative of the cost function and solving for zero will provide us with a new equation for finding optimal β values

$$\hat{\beta}_{\text{Ridge}} = \left(X^TX + \lambda I\right)^{-1}X^Ty.$$

The regularization parameter λ affects the optimal β value, which we will study the effects off by using these new equations and calculating the MSE. This is similar to the OLS process.

Lasso regression

Lasso regression is the third and last regression method used in this study. Like ridge regression, lasso also includes the regularization parameter λ and is built semi-similar to the other regression methods. The cost function for lasso regression is

$$C(X,\beta) = \frac{1}{n}\left\{(y - X\beta)^T(y - X\beta)\right\} + \lambda||\beta||_1.$$

Finding the derivative of the cost function with regards to β and solving for zero, will **not** yield a neat analytical equation like for the previous regression methods. The resulting equation is as follows

$$X^TX\beta + \lambda\frac{d|\beta|}{d\beta} = 2X^Ty.$$

Since the result isn't a closed-form expression, we will utilize pythons **sklearn** lasso linear model when finding the MSE with lasso regression.

Bootstrap resampling

Bootstrap is a resampling method that resamples random values in our training data and replaces it in a new data set. Subsequently using the new data set to find $\hat{\beta}$ and MSE values. The resampling is repeated n_b amount of times, then the mean values for MSE will be used in the chosen regression method.

K-fold cross-validation

K-fold cross-validation is another resampling method that resamples our data and divides it into k parts. One of the k parts are then used as test data, while the rest is used as training data for the regression method. This resampling will run k amount of times such that all the k-folds have been used as test data. As for the bootstrap method, the mean MSE values will be used as the error for the chosen regression model. The aim of cross-validation is to get a wider selection of test and training data.

III. Results & Discussion

Scaling

When doing regression for the Franke function we employ $x, y \in [0, 1]$ which in turn produces z values that are similar. The product of our data set should, in this case, not need scaling, due to it already being scaled down to some degree. Regardless, the data set was still scaled using pythons package Standardscaler from the **sklearn** library. If the data wasn't scaled, it could produce inaccurate results, especially for ridge and lasso regression. Since ridge and lasso have coefficients that are managed by the regularization parameter, not scaling the data could lead to a bigger accuracy hit than for OLS. Scaling wasn't strictly necessary for OLS, but since it was needed for ridge and lasso, it was also applied for OLS for consistency and a more accurate comparison between the methods.

While studying the terrain data, scaling was also applied. Since the terrain data didn't have pre-scaled x, y values, it would result in flawed results if left unscaled.

a. The Franke function

Plots for the MSE and R^2-score from using OLS without resampling are as shown in figure 1 and figure 2. From these plots it's clear to see a trend of the MSE lowering towards 0 as the polynomial degree escalates, showing that OLS predicts a more accurate result for higher polynomial degrees. The plot of R^2-score against polynomial degree shows a negative R^2-score which means that the predicted values are worse than if the average predicted value was used instead.

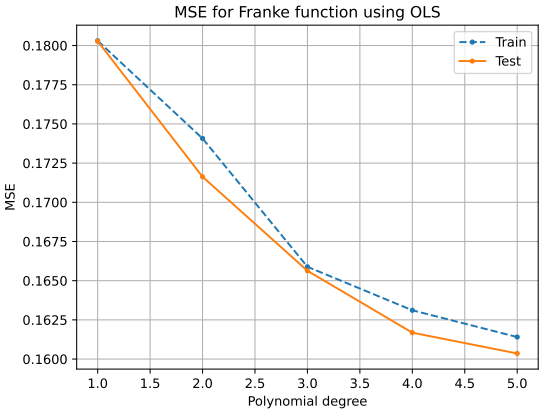

Fig. 1: Plot of the MSE against polynomial degree using OLS and the Franke function. The stochastic noise ε is given by the normal distribution $N(0, 0.1)$. Number of samples $n = 100$.

Figure 3 plots the coefficients β evolution for each polynomial degree. For the lower polynomial degrees, the coefficient is close to zero and quite stable, however, for the higher degrees there are big variations of coefficients. Variations like these happen because the regression method attempts to fit the model for a higher order polynomial. If the variation would increase for higher degrees, it would indicate that the model is over-fitting. For a polynomial degree of five, the coefficient seems to start of with big variations and then get lower variations later on in the model,

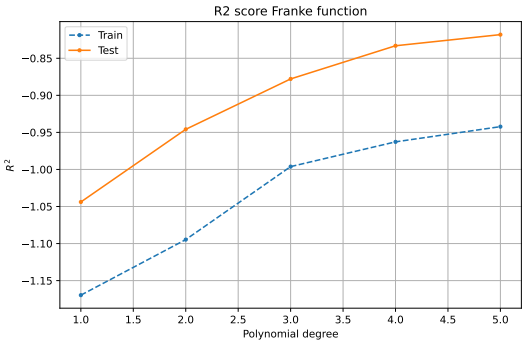

Fig. 2: R^2-score plotted against the polynomial degree for the Franke function.

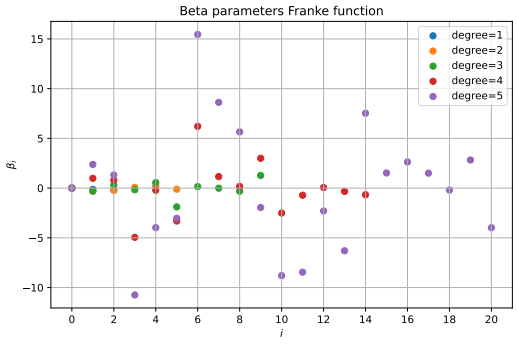

Fig. 3: Plot of the coefficients as polynomial degrees using the Franke function.

indicating that the model isn't over-fitting at this degree.

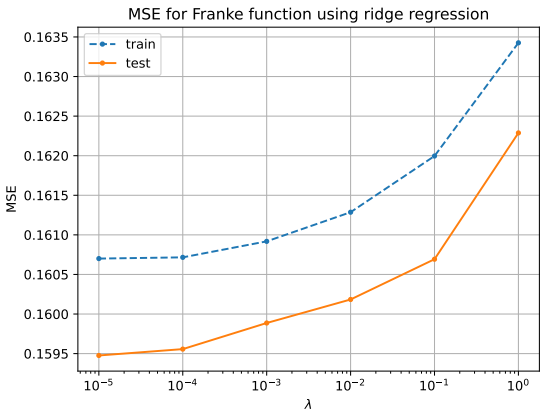

Fig. 4: Plot of MSE for polynomial degree 6 and different λ values using ridge regression and the Franke function. Number of samples $n = 100$.

Figure 4 shows the MSE plotted against different values of λ. It seems that OLS and ridge regression have roughly the same MSE when OLS uses a higher polynomial degree and ridge uses lower values of λ. This makes sense because λ keeps the coefficient β from deviating vigorously.

Our plotted MSE against polynomial degree with a set λ using lasso regression is depicted in figure 5. The MSE that the lasso regression gives is noticeably lower than for

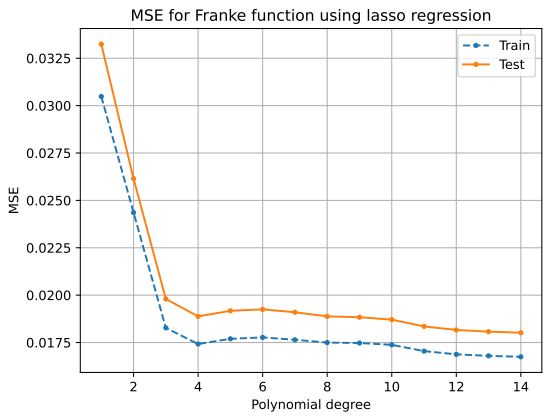

Fig. 5: Plot of MSE for polynomial degree 15 and $\lambda = 0.001$ using lasso and the Franke function. Number of samples $n = 100$.

both OLS and ridge. Currently this is the most accurate regression method.

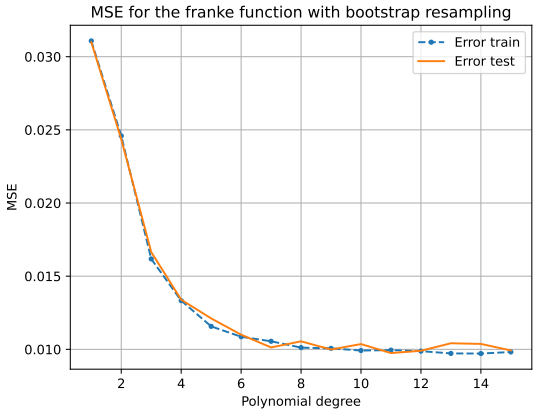

Fig. 6: Plot of MSE for different polynomial degrees using OLS with bootstrap resampling and the Franke function. Number of samples are $n = 100$ and $n_b = 100$.

Figure 6 shows MSE plotted against polynomial degree for OLS with bootstrap resampling applied. The resampling has lowered the MSE significantly from figure 1, which is OLS without any resampling. The same pattern still applies, lower values of MSE for higher polynomial degrees.

The bias-variance tradeoff plotted in figure 7 shows a variance very close to zero and a bias which is as close to the calculated error as can be. If bias and variance are added together it would result in them following the error perfectly. This relation between bias, variance and error proves a sound implementation of the OLS method with bootstrap resampling. A low variance and a higher bias means that our model is under-fitting our data.

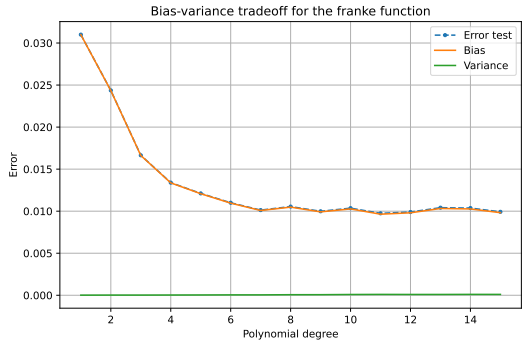

Fig. 7: Plot of bias-variance tradeoff against polynomial degree using OLS, bootstrap and the Franke function. Number of samples are $n = 100$ and $n_b = 100$.

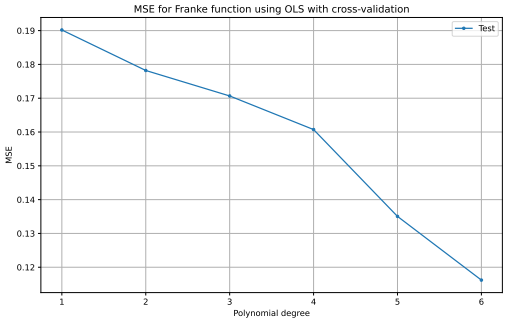

Fig. 8: Plot of MSE for different polynomial degrees using OLS with k-fold cross-validation and the Franke function. Number of samples $n = 50$ and number of folds $k = 10$.

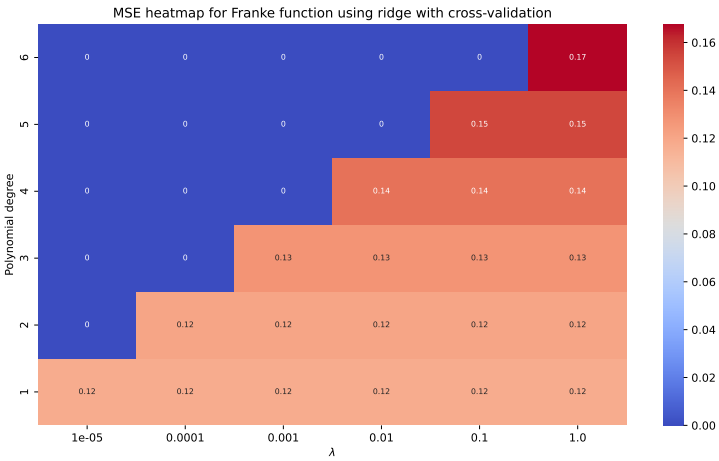

Fig. 9: Heatmap plot of MSE for different polynomial degrees and λ using ridge with k-fold cross-validation and the Franke function. Number of samples $n = 50$ and number of folds $k = 10$.

Implementation of the cross-validation resampling technique provides plots for OLS, ridge and lasso in figures 8, 9 and 10. MSE for OLS with cross-validation is shown to be significantly lower than without, as seen in figure 1 . In figure 9 the MSE is observed to be low for low polynomial degrees and low values of λ. Compared to the MSE provided from ridge regression in figure 4, without cross-validation, our new predictions seem to be more accurate than the former for all λ values and polynomial degrees.
The MSE from our lasso regression with cross-validation, as

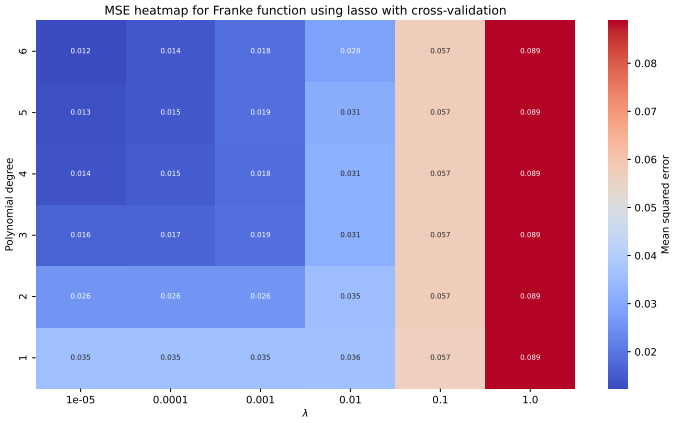

Fig. 10: Heatmap plot of MSE for different polynomial degrees and λ using lasso with k-fold cross-validation and the Franke function. Number of samples $n = 50$ and number of folds $k = 10$.

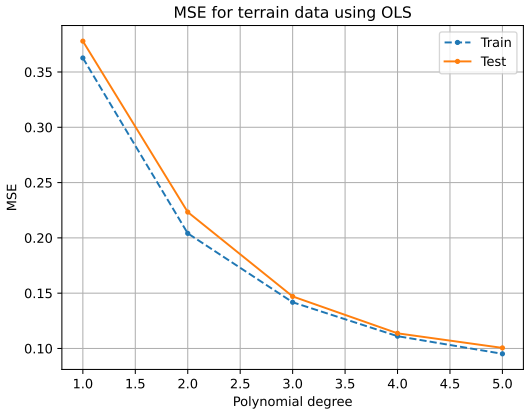

Fig. 11: Plot of the MSE against polynomial degree using OLS and the terrain data. The stochastic noise ε is given by the normal distribution $N(0, 0.1)$ and number of samples are $n = 100$.

seen in figure 10, has no exceptional response to the resampling method. The MSE is almost unchanged from the ones we see without cross-validation in figure 5. It has actually estimated a higher MSE for the same λ, which was set to 0.001 in the lasso regression without resampling.
Evidently the implementation of k-fold cross-validation shows improvements in the predictions of the Franke function data set for both OLS and ridge regression. For lasso regression on the other hand, it seems to stay mostly unchanged, but with a minor increase in MSE with the implementation of cross-validation.

b. The terrain data

Plots of MSE against polynomial degree after employing OLS without resampling on the terrain data are as shown in figure 11. There is an obvious trend of lower MSE for higher polynomial degrees. As for the Franke function this trend shows improvement for OLS regression accuracy using higher polynomial degrees. MSE for the terrain data starts of at higher values and ends lower than for the Franke function, which is probably due to internal differences for the data sets. The R^2-score plotted in figure 12 shows appreciably better results than for the Franke function. R^2-score curves are positive, the training and test data are close to each other and the curve is climbing towards $R^2 = 1.0$. The results imply a model which has prominent accuracy for the predicted values.

The coefficients β plotted in figure 13 see the same pattern as for the Franke function. Higher variations in coefficients for higher degrees, with bigger variations in the beginning of the fitting process. This implies that the method isn't over-fitting at these low polynomial degrees for the terrain data either.

MSE for ridge regression in figure 14 shows an increase in MSE for higher values of λ, showing the regressions dependency on a low regularization parameter. For both low and high values of λ, the MSE is still roughly the same as for OLS at higher polynomial degrees.

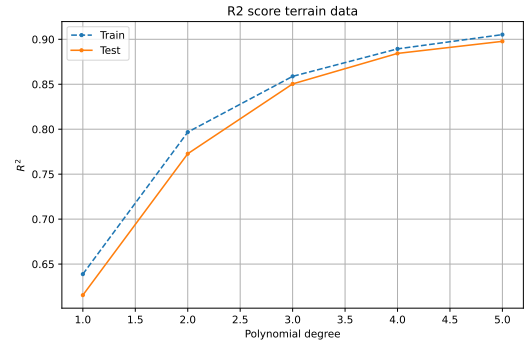

Fig. 12: Plot of the R^2-score against the polynomial degree using OLS and the terrain data.

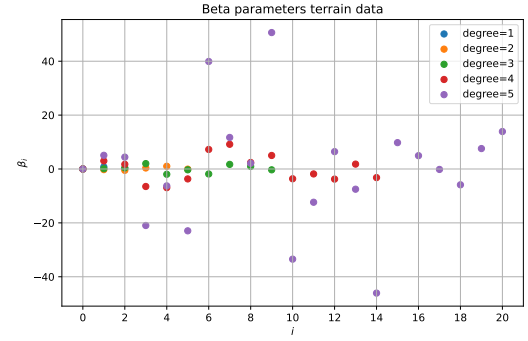

Fig. 13: Plot of the coefficients as polynomial degrees using the terrain data.

When implementing the lasso regression with $\lambda = 0.001$ we got the MSE shown in figure 15. The MSE never reaches 0.1, not even for the higher polynomial degrees. For the terrain data analysis, lasso regression has performed the worst out of the three regression methods.

Figure 16 shows MSE plotted against polynomial degree for OLS with bootstrap resampling applied. Bootstrap resampling doesn't seem to improve the model much. Figure 16 indicates both the same MSE for lower polynomial degrees and for higher polynomial degrees as the OLS analysis done without resampling in figure 11.

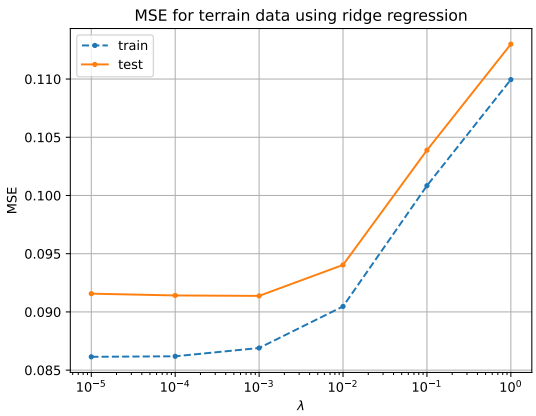

Fig. 14: Plot of MSE for polynomial degree 6 and different λ values using ridge regression and the terrain data.

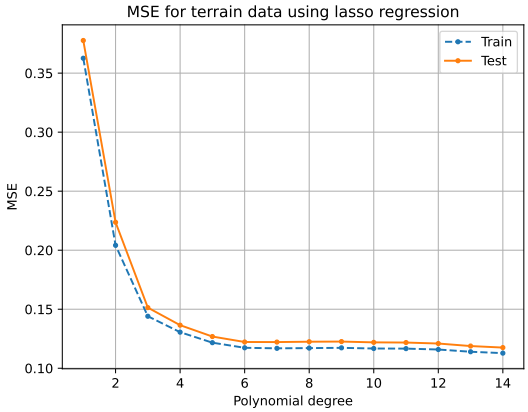

Fig. 15: Plot of MSE for polynomial degree 15 and $\lambda = 0.001$ using lasso regression and the terrain data.

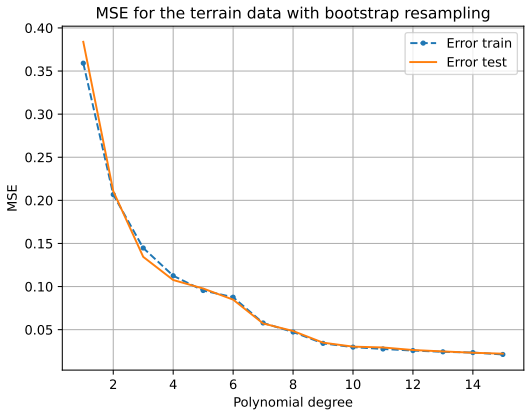

Fig. 16: Plot of MSE for different polynomial degrees using OLS with bootstrap resampling and the terrain data.

Figure 17 shows the bias-variance tradeoff for OLS with bootstrap resampling. Similar to the same study done on the Franke function, our model has a bias almost coinciding with the error and a variance which is almost zero. Bias and variance added together would follow the error perfectly, which indicates that our model is under-fitting our data. The relation between bias, variance and error proves a sound implementation of OLS with bootstrap resampling.

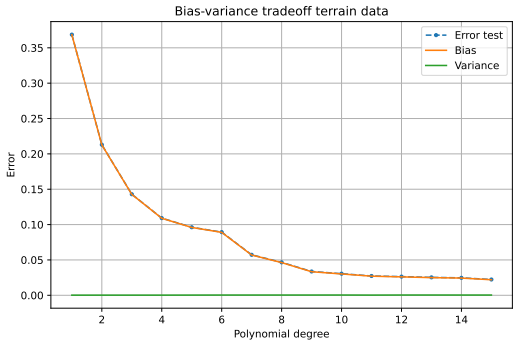

Fig. 17: Plot of the error, bias and variance against polynomial degree with OLS and bootstrap resampling using the terrain data.

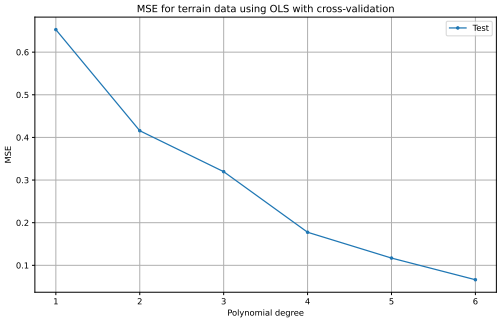

Fig. 18: Plot of MSE for different polynomial degrees using OLS with k-fold cross-validation and the terrain data. Number of samples $n = 50$ and number of folds $k = 10$.

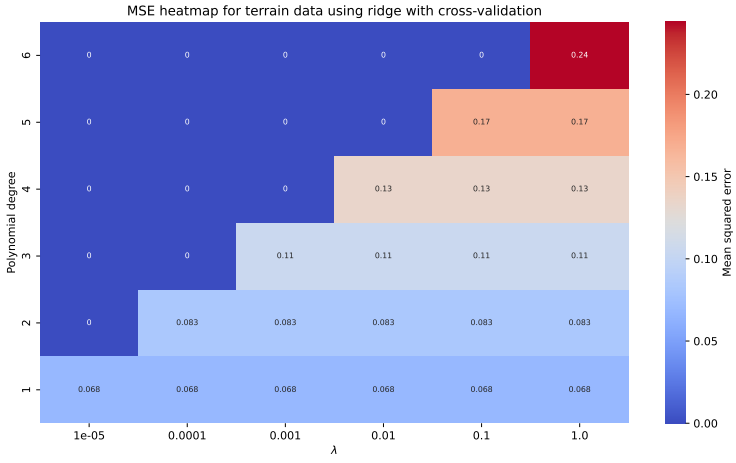

Fig. 19: Plot of MSE for different polynomial degrees using ridge with k-fold cross-validation and the terrain data. Number of samples $n = 50$ and number of folds $k = 10$.

All three regression methods with implementation of k-fold cross-validation give the results seen in figures 18, 19 and 20. MSE for OLS in figure 18 provides about the same results as for bootstrap resampling and OLS without resampling. Neither resampling method seems to provide an improvement to our OLS regression.
For ridge regression we see a lower valued MSE for $\lambda < 1$ from the OLS results. With the lowest plotted values of λ, figure 19 shows a very low MSE. As soon as the MSE reaches below 0.01 it shows a clean zero instead, which indicates a well-working ridge regression for the terrain data.

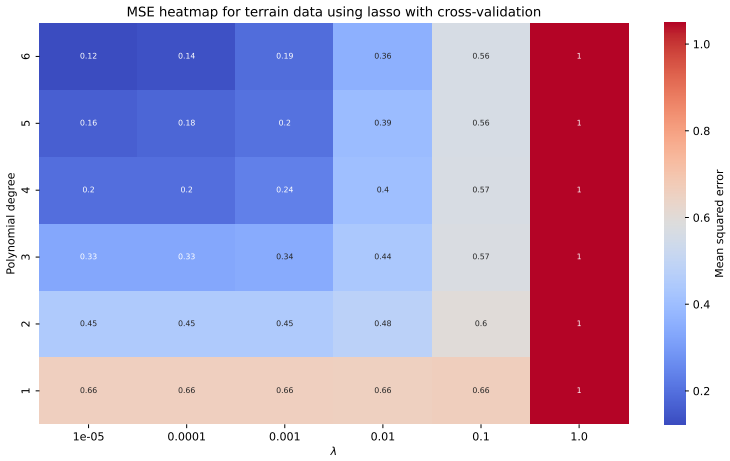

Fig. 20: Plot of MSE for different polynomial degrees using lasso with k-fold cross-validation and the terrain data. Number of samples $n = 50$ and number of folds $k = 10$.

MSE for lasso regression are quite high across figure 20. Not until the lowest plotted λ value does lasso regression compare to the MSE of OLS. However, the pattern is clear to see. Higher values of polynomial degrees and lower values of λ provide the lowest MSE and the best result. Lasso regression seems to be doing the worst out of the three regression methods when employing them on the terrain data. Additionally, considering that lasso regression needs significantly more computational power to use, ridge regression would be much more accurate and efficient.

IV. Conclusion

By succesfully employing OLS, ridge and lasso regression to the Franke function and the terrain data set, we are now able to distinguish which method predicts the best results. Analyzing how our results were affected by using bootstrap and k-fold cross-validation, the results have been varied from each data set. The Franke function data set had a great response to both resampling methods, while the terrain data didn't differ for either resampling method. In conclusion, it is important to select the appropriate regression method based on the data set.

APPENDIX A DERIVING BIAS AND VARIANCE

Assuming that there exists a continuous function $f(x)$ and a normal distributed error $\varepsilon \sim N(0,\sigma^2)$ that describes our data as

$$y = f(x) + \varepsilon,$$

and that the function f is approximated by $\tilde{y}$ where we wish to minimize the difference $(y-\tilde{y})^2$[2]. We get

$$\tilde{y} = X\beta.$$

Then we can calculate the expectation value (bias) $\mathbb{E}(y_i)$ as follows:

$$\mathbb{E}(y_i) = \mathbb{E}(\mathbf{X}_{i,*}\beta) + \mathbb{E}(\varepsilon_i) \;=\; \mathbf{X}_{i,*}\beta.$$

Variance for y for a given element i is

$$\begin{aligned}
\mathrm{Var}(y_i) = \mathbb{E}\{[y_i - \mathbb{E}(y_i)]^2\} \;&=\; \mathbb{E}(y_i^2) - [\mathbb{E}(y_i)]^2 \\
&= \mathbb{E}[(\mathbf{X}_{i,*}\beta + \varepsilon_i)^2] - (\mathbf{X}_{i,*}\beta)^2 \\
&= \mathbb{E}[(\mathbf{X}_{i,*}\beta)^2 + 2\varepsilon_i\mathbf{X}_{i,*}\beta + \varepsilon_i^2] - (\mathbf{X}_{i,*}\beta)^2 \\
&= (\mathbf{X}_{i,*}\beta)^2 + 2\mathbb{E}(\varepsilon_i)\mathbf{X}_{i,*}\beta + \mathbb{E}(\varepsilon_i^2) - (\mathbf{X}_{i,*}\beta)^2 \\
&= \mathbb{E}(\varepsilon_i^2) \;=\; \mathrm{Var}(\varepsilon_i) \;=\; \sigma^2.
\end{aligned}$$

Expectation values (bias) for the optimal value $\hat{\beta}$ are calculated as follows

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{Y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta = \beta.$$

The variance for the optimal value $\hat{\beta}$ is calculated like this

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}) \;&=\; \mathbb{E}\{[\beta - \mathbb{E}(\beta)][\beta - \mathbb{E}(\beta)]^T\} \\
&=\; \mathbb{E}\{[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} - \beta]\,[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} - \beta]^T\} \\
&=\; (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\,\mathbb{E}\{\mathbf{Y}\mathbf{Y}^T\}\,\mathbf{X}\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \\
&=\; (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\,\{\mathbf{X}\beta\,\beta^T\mathbf{X}^T + \sigma^2\}\,\mathbf{X}\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \\
&=\; \beta\,\beta^T + \sigma^2\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \;=\; \sigma^2\,(\mathbf{X}^T\mathbf{X})^{-1}.
\end{aligned}$$

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] "Week 36: Statistical interpretation of linear regression and resampling techniques," (2023 September), lecture notes. [Online]. Available: https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/week36.html,(Visited15.10.2023)