FYS-STK4155 - Project 1

Linear regression analysis and resampling methods

Fredrik Hoftun & Ada Weinert Ravn
University of Oslo

(Dated: January 4, 2022)

I. INTRODUCTION

Regression analysis dates back to the early 19th century, where the mathematicians Legandre and Gauss independently developed the method of least squares in order to determine planetary orbits from observations. Today, regression is used in pretty much every field which deals with prediction or forecasting. Regression aims to reveal relations between a dependent variable and a collection of independent variables in a dataset.
The goal of this project is to explore the performance and applicability of different linear regression methods with different resampling methods. We will implement Ordinary Least Squares (OLS), Ridge and Lasso regression models to fit polynomials to specific 2D sets of data. The models will be tested on the Franke's function, a function widely used for testing various fitting and interpolation algorithms, and later applied to real terrain data. Proper assessment of our models will be performed using bootstrap resampling, where we will also discuss the bias-variance trade-off and R^2 score of each model. Comparing the application of different methods on the same models will give us an opportunity to evaluate the relative performance and precision of each of them.

We will first describe the methods and models used. The results, including comparison of different models, will be presented afterwards followed by a short discussion. Finally we will finish with a conclusion based on our analysis.

II. THEORY

Before we proceed, we need to introduce a few concepts centered around our project. The scope of this project will be defined by the Ordinary Least Squares (OLS), Ridge and Lasso linear regression models applied using the Bootstrap and k-fold Cross-validation. Bias-variance trade-off and coefficient of determination will be discussed as a measure of model applicability.

A. Linear regression

Regression is a method aiming to fit a set of measured data $y = [y_0, y_1, y_2, ..., y_{n-1}]$ to a statistical model described by some unknown function f with $x = [x_0, x_1, x_2, ..., x_{n-1}]$ as inputs, i.e. construct a function $y(x)$ where $y_i = y(x_i)$.

Assuming the linear relation between the dependent variable y and the regressor vector x. The relationship is modelled with addition of error variable ϵ and takes the following form

$$y = y(x) \rightarrow y(x_i) = \hat{y} + \epsilon = \beta_0 + \beta_1 x + \epsilon$$

In many cases, including surface interpolation, the linear relationship might not hold. As a result we need to expand this solution into the general quadratic model of the form

$$y = y(x) \rightarrow y(x_i) = \hat{y_i} + \epsilon_i = \sum_{j=0}^{n} \beta_j x_i^j + \epsilon_i$$

Here β_j is an unknown parameter, ϵ_i is the error of our prediction $\hat{y_i}$ and n is the degree of our polynomial. This results in following set of equations for every pair of values y_i, x_i

$$y_0 = \beta_0 + \beta_1 x_0^1 + \beta_2 x_0^2 + ... + \beta_{n-1} x_0^{n-1} + \epsilon_0$$

$$y_1 = \beta_0 + \beta_1 x_1^1 + \beta_2 x_1^2 + ... + \beta_{n-1} x_1^{n-1} + \epsilon_1$$

$$y_2 = \beta_2 + \beta_1 x_2^1 + \beta_2 x_2^2 + ... + \beta_{n-1} x_2^{n-1} + \epsilon_2$$

$$\vdots$$

$$y_{n-1} = \beta_0 + \beta_1 x_{n-1}^1 + \beta_2 x_{n-1}^2 + ... + \beta_{n-1} x_{n-1}^{n-1} + \epsilon_{n-1}$$

The problem can be then solved by rewriting it as a set of vectors along the design matrix X

$$y = [y_0, y_1, y_2, ..., y_{n-1}]^T$$

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

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

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

Giving us the following equation for our function

$$y = y(x) \rightarrow y(x_i) = \hat{y} + \epsilon = X\beta + \epsilon$$

1. Ordinary Least Squares regression

Among the linear regression models the most popular one is the Ordinary Least Squares where we pick parameters β so that the distance between our target y and the prediction $\hat{y}$, and hence the error ϵ, are minimized. This is also known as the mean squared error (MSE), which we then use to define our cost function

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

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

The minima of the cost function can be found by taking the derivative with respect to β and setting it to be equal to zero

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

$$X^T y = X^T X\beta$$

Assuming that the matrix $X^T X$ is invertible we get the optimal values for the parameters β

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

The predicted values of $\hat{y}$ are then given by

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

2. Ridge and Lasso Regression

The Ordinary Least Squares model my be further expanded to improve its performance. The Ridge regression introduces a new parameter in its cost function - the regularization penalty $\lambda >= 0$. The resulting expression is as follows

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

Similarly to the solution for OLS we find the minima of the cost function and find that

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

Where I is a $p \times p$ identity matrix with the constraint that

$$\sum_{i=0}^{p-1}\beta_i^2 <= t$$

for a finite, positive t. The predicted values of $\hat{y}$ are given by

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

Another method implementing the regularization penalty λ is the Lasso (Least Absolute Shrinkage and Selection Operator) method, with the following cost function

$$C(\beta) = \{(y - X\beta)^T(y - X\beta)\} + \lambda\sqrt{\beta^T\beta}$$

The solution to the minima of the cost function is comparable to the Ridge solution

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

This gives the following predicted values of $\hat{y}$

$$\hat{y} = X\beta = X(X^T X + \lambda\sqrt{I})^{-1}X^T y$$

For each of those models if the regularization penalty $\lambda = 0$ the cost equation and following prediction will revert into the Ordinary Least Squares solution. For $\lambda > 0$ a constraint is added to the coefficient, decreasing the value of it and causing it to tend towards zero.

The difference between the Ridge and Lasso regression models is that the increase in the penalty term tends to drive more and more parameters to zero in Lasso regression, while in Ridge regression the parameters are reduced while remaining non-zero. As a result Lasso regression model is suitable for feature reduction, as it selects more relevant features and discards the others. The Ridge model reduces the complexity of the model but not overall feature count.

We will apply both of those methods in our surface interpolation problem and evaluate their performance.

B. Assessing model accuracy

1. Estimators

An estimator is an algorithm for calculating an estimate from a data set. Popular estimators are error, mean/average, variance, bias and the MSE. Another popular method to determine the quality of a data set is the coefficient of determination or R^2, which shows how well the data set fits the predicted data set in the regression model.
The MSE is defined as

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

and the R^2 score as

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

where $\bar{y}$ is the mean value of y.

2. Bias-variance trade-off

The art of machine learning is creating accurate models with high accuracy. A common way to assess model accuracy is through the MSE. The cost function of the MSE is

$$C(\beta, X) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = E[(y_i - \hat{y}_i)^2]$$

We can further work at the expression

$$= E[y_i - \hat{y}_i + E[\hat{y}_i] - E[\hat{y}_i])^2]$$

$$= E[(y_i - E[\hat{y}_i])(E[\hat{y}_i] - \hat{y}_i))^2]$$

$$= E[(y_i - E[\hat{y}_i])^2 + 2((y_i - E[\hat{y}_i])(E[\hat{y}_i] - \hat{y}_i)) + (E[\hat{y}_i] - \hat{y}_i)^2]$$

$$= E[(y_i - E[\hat{y}_i])^2] + E[2((y_i - E[\hat{y}_i])(E[\hat{y}_i] - \hat{y}_i))] + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

$$= E[(y_i - E[\hat{y}_i])^2] + 2(y_i - E[\hat{y}_i])(E[E[\hat{y}_i] - \hat{y}_i]) + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

We see that the second part of our expression equals zero since $E[E[\hat{y}_i]] - E[\hat{y}_i] = 0$. So we now have

$$E[(y_i - E[\hat{y}_i])^2] + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

Real y_i values will however have some noise $y_i = f(x_1, x_2, ..., x_n) + \epsilon$. If we assume the noise is normally distributed with a mean equal to zero we get

$$E[(f(x_i) + \epsilon - \hat{y}_i)^2]$$

And our expression becomes

$$E[(f(x_i) + \epsilon - E[\hat{y}_i])^2] + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

$$= E[f(x_i)^2 + \epsilon^2 + E[\hat{y}_i]^2 + 2f(x_i)\epsilon - 2f(x_i)E[\hat{y}_i]$$

$$-2\epsilon E[\hat{y}_i]] + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

But $E[\epsilon] = 0$ and $E[\epsilon^2] = \sigma^2 = Var(\epsilon)$. Then we have

$$= E[f(x_i)^2 + E[\hat{y}_i]^2 - 2f(x_i)E[\hat{y}_i]] + Var(\epsilon) + E[(E[\hat{y}_i] - \hat{y}_i)^2]$$

$$= E[(f(x_i) - E[\hat{y}_i])^2] + Var(\epsilon) + E[(\hat{y}_i - E[\hat{y}_i])^2]$$

The first moment is called the squared bias of the predicted values, the second moment is the variance of the noise and the third moment is the variance of the predicted values.

$$= Bias(\hat{y}_i)^2 + Var(\epsilon) + Var(\hat{y}_i)$$

Bias is the errors from assumption, e.g. we assume that the true form of a model is linear. The greater the bias the stronger the assumptions of the model. Variance is the average of the squared deviation, or error, from the true values. So in order to minimize the MSE we need to minimize the bias and variance of the model. The variance of the noise is given by the data set and can't be changed.
Intuitively, having high bias mean our model will have low variance, as we have assumed much of the model, which means our predicted errors will be smaller. Likewise, if we have low bias, we will have higher variance. This property is called the *bias-variance trade-off*.A big part of finding the best model for a given data set is to balance this bias-variance trade-off, finding the right amount of assumptions to make in order for the variance to be acceptable.

3. Resampling

Resampling is a technique used in statistics to estimate the quality of a sample, and validate the predictive ability of a model. Two popular resampling methods are the bootstrap and k-fold cross-validation methods.
The bootstrap method picks random samples from a training set and and adds them to a test set. This goes on for however many *bootstraps* you want to do. Since the samples are picked randomly this allows for the same sample to be picked more than once. This is called sampling with replacement. When the bootstrap is done, the statistical properties of the model can be measured using the test set and an estimator, like the MSE or variance. The advantage of the bootstrap is that it is useful on random distributions with small sample sizes, as the statistical properties of the distributions can be checked by bootstrapping the distribution.
k-fold cross validation works by dividing the training set into k subsets of equal size and using one of them as a test set. We then iterate over the subsets k times and swap the test set for another unused subset, so that when the iteration is over, every subset has been used as a test set. We can then measure the statistical properties of the test sets with an estimator. Cross validation is useful when checking if the model has been overfitted, as the MSE in the test set will exceed it's anticipated value.

III. DATASETS

We will first test our implementation on a specific two-dimensional function, called Franke function. The established models and methods will then be applied to real digital terrain data, to try to reproduce it.

A. The Franke Function

The Franke function is a test function commonly used to evaluate different surface interpolation techniques. It consists of a surface with "twin Gaussian peaks and a sharper Gaussian dip superimposed on a surface sloping towards the first quadrant"[2]. The function, given by a weighted sum of exponentials, is evaluated for $x, y \in [0, 1]$, with the addition of stochastic noise ϵ generated from a normal distribution $N(0, 1)$. The full form of the Franke function, as seen in Figure 1 is given as

$$f(\mathbf{x}, \mathbf{y}) = \frac{3}{4}\exp\left\{-\frac{1}{4}(9x-2)^2 - \frac{1}{4}(9y-2)^2\right\}$$

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

$$+\frac{1}{2}\exp\left\{-\frac{1}{4}(9x-7)^2 - \frac{1}{4}(9y-3)^2\right\}$$

$$-\frac{1}{5}\exp\{-(9x-4)^2 - (9y-7)^2\}$$

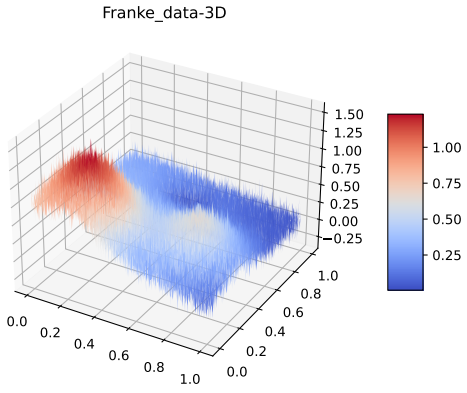


Figure 1. The Franke function, for $x, y \in [0, 1]$ with added stochastic noise ϵ

B. Digital terrain data

The terrain data used was chosen from options available in the course Github repository[3], representing the area near Stavanger, Norway. The data was originally taken form the U.S. Department of the Interior U.S. Geological Survey's (USGS) EarthExplorer[4] data in SRTM Arc-Second Global format. A $N \times N$ slice has been cut out from the terrain data for the purpose of testing our terrain parametrization. The visual representation of it can be seen in Figure 2 and 3.

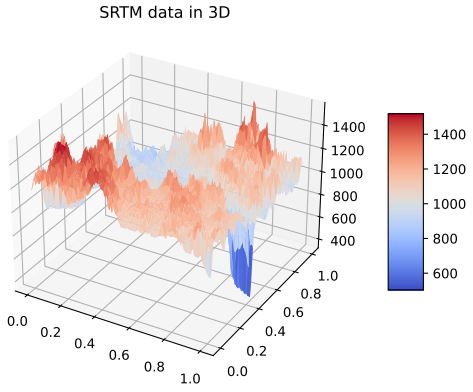


Figure 2. Terrain over Stavanger, Norway 3D view.

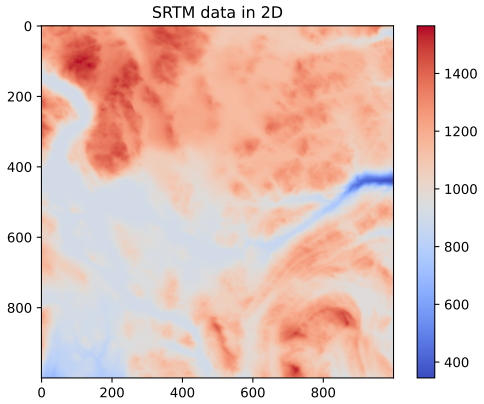


Figure 3. Terrain over Stavanger, Norway 2D view.

IV. IMPLEMENTATION

The ols, ridge and lasso models have been implemented manually and with functions from the Sci-Kit Library [1]. All of the outputs presented in this graph have been produced using the built in methods, other than the lasso regression model. The dataset has been scaled since the data is expected to behave strangely if individual features do not more or less look like standard normally distributed data. In this case the scaling was performed by the means of removal of the mean. We have chosen to split the data 8/2 between the training and testing set, and use the value $sigma = 0.05$ to represent noise for all tasks using the Frank function.

[1] Link to documentation: https://scikit-learn.org/stable/modules/classes.html

V. RESULTS AND DISCUSSION

Only a couple of example plots are included for each of the problems, all figures otherwise can be found in the GitHub repository and as such will often be referenced to avoid clutter. Please note that N denotes the size of one axis, not total amount of data points.

A. Confidence intervals

The base ols algorithm is run for N=20 samples, sigma=0.03 and max_degree=5. The resulting value of R-squares score is 0.992096 and of MSE 0.000649. Even in the blind we can consider those to be great scores. We notice that for the tested degrees the metrics improve with increase in the degree.

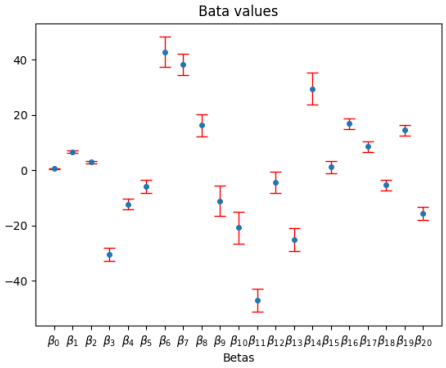


Figure 4. 95% confidence interval for β parameters, max_degree=5, N=20.

Next we plot the beta values for the same selection of parameters. The output can be seen in Figure 15. Additional test indicate that that the interval size decreases for higher N counts.

B. OLS

We will first take a look at results achieved using the basic OLS algorithm..

Before discussing the bias-variance trade-off we will plot MSE scores for the training and testing against each other to identify regions of high and low bias/variance. The output can be seen in Figure 5.

While the value of the MSE_train steadily decreases, MSE_test begins to shoot up around 11 degree mark. The reason for that is that a model with higher complexity will be more tailored to the data,as a result reacting poorly to test data and resulting in overfitting. We identify potential regions of convergence around degree 5, 7 and 10 for N=20. We plot the bias-variance trade-off graph for the same parameters and compare them. The output can be seen in Figure 6 where it is clear that there

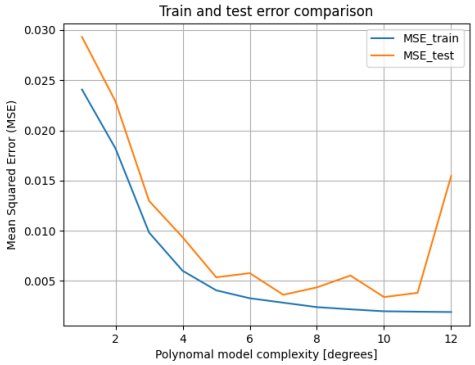


Figure 5. Comparison of training and testing error, N=20.

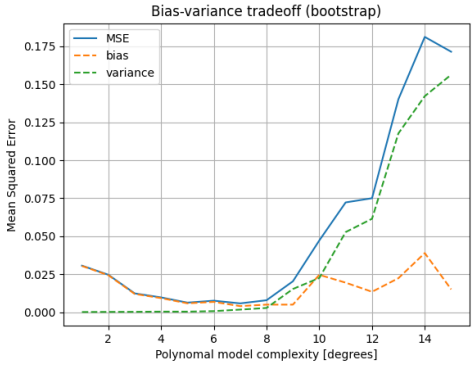


Figure 6. Bias-variance trade-off, OLS, N=20.

exists a correlation between the regions of convergence in each graph based on their relative proximity. We can see that our expectation of the predicted convergence points has been correct, especially around the 10 degree mark. It is also around there that a difference in the composition of the error appears. Before that point bias made up the majority of the error score, and past it it is replaced with variance as the top contributor.

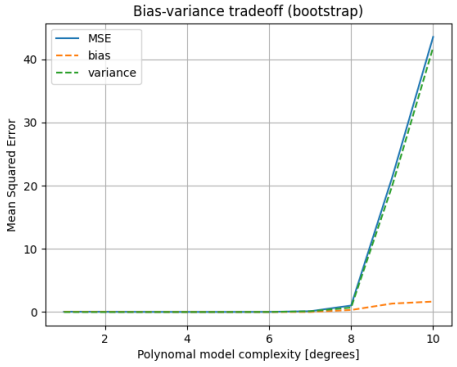


Figure 7. Bias-variance trade-off, OLS, N=10.

Still exploring the bootstrap of the OLS algorithm we

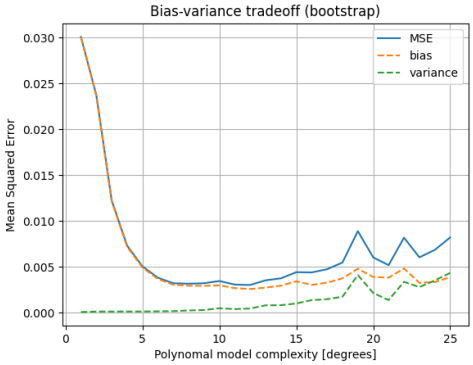


Figure 8. Bias-variance trade-off, OLS, N=30.

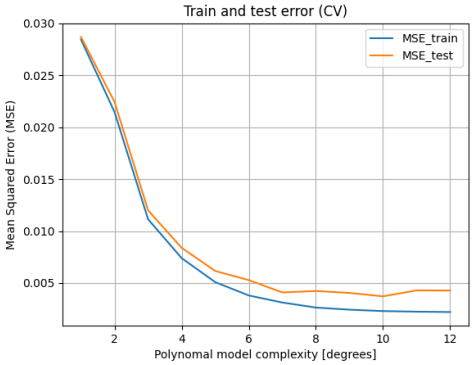


Figure 10. MSE train and test data OLS, CV, N=20.

a look at the bias-variance decomposition with regards to both complexity and the number of elements. Based on Figures 7, 6 and 8 we can see that the optimal fit point at the crossing of bias and variance changes with regards to model complexity - resulting in higher degree matching higher number of elements. This is seen as best fit for each of them and is as follows degree=4 for N=10, degree=8 for N=20 and degree=23 for N=30. In other words models with higher number of input element are more robust to overfitting with increase in complexity.

Next we will look at the performance of the k-means cross-validation algorithm and compare it to the outputs from bootstrap. To find optimal number of folds to be used we evaluate the output according to the number of folds applied. It is clear that the MSE of the test set decreases with increasing number of folds, even at higher complexities. This can be seen in resulting figures - found in the GitHub repository.

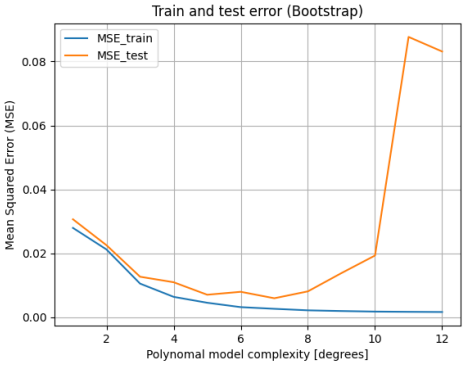


Figure 9. MSE train and test data OLS, bootstrap, N=20.

Next we will compare performance of bootstrap and K-means resampling methods respectively for varying amounts of input variables N. We will from now use 5 folds in cross validation and 50 bootstraps for the k-fold cross-validation and bootstrap algorithms respectively. Again the complete list of figures can be found in the GitHub repository, the Figures 9 and 10, estimated for N=20, represent the overall trend of the findings - for higher model complexities and higher amount of input variables the k-fold cross-validation performs significantly better than the bootstrap.

C. Ridge

Now we are going to take a look at the Ridge algorithm. We will preform the bias-variance trade-off analysis with regards to λ and compare both of the resampling methods once again. We will use number of variables of N=20 here.

We chose a selection of three λ values from the tested images, as shown in Figure 11. It is apparent that the error decreases with lower values of λ. Interestingly though for higher model complexities the variance begins to increase while using particularly small values of λ. Otherwise its contribution to the total error is minuscule.

Following we have a comparison of the resampling method outputs where once again the k-fold cross-validation resampling method performs much better than the bootstrap we will take a look at λ values of $\lambda = 0.0001$, $\lambda = 0.01$ and $\lambda = 1$ in Figure 12.

We can see that once again that the k-means cross-validation resampling performs much better than the bootstrap and is less prone to overfitting. When it comes to influence of the λ parameter itself it appears to significantly improve the accuracy of prediction, especially with very low values.

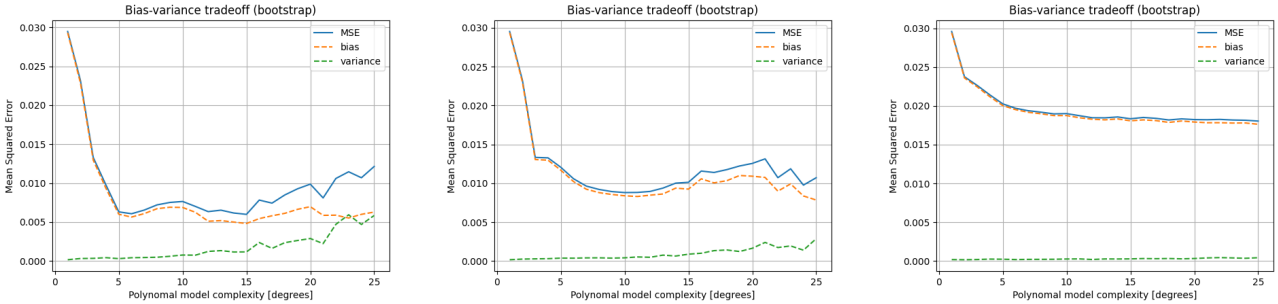
D. Lasso

Now we are going to take a look at the Lasso algorithm. We will preform the bias-variance trade-off analysis with regards to λ and compare both of the resampling methods once again. We will use number of variables of N=20 here.

We immediately notice that even lower values for λ might be needed to properly evaluate the performance of

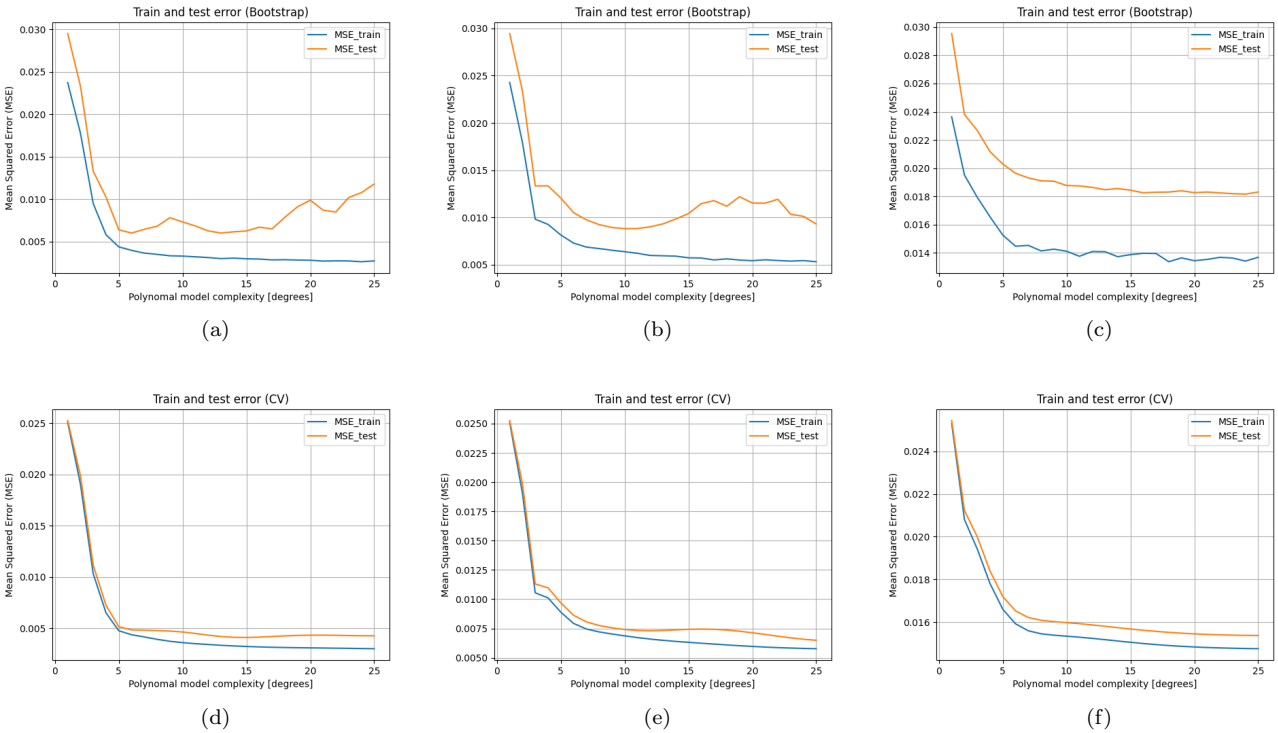
the algorithm, as seen in the bias-variance trade-off graph in Figure 21. Due to the time constraints only values as low as 0.001 were implemented, and the same λ inserted into the Ridge Algorithm performs much better, and with a lower variance. Though we notice that the MSE score improves with lower λ values as was the case in the Ridge implementation. Otherwise we notice that once again bias makes up the vast majority of the total error, just like in the case of the OLS and Ridge algorithms.

Following we have a comparison of the resampling method outputs where once again the k-fold cross-validation, here once again it feels like smaller values of λ need to be tested to get a proper overview over the performance of the model. Still we will take a look at λ values of $\lambda = 0.001$ and $\lambda = 0.01$ in Figure 22. In the case of the bootstrap algorithm we can see that the decrease of λ leads to improvement in the model, but the for cross-validation all we can realistically say that the values we were able to test do not provide a reasonable solution.



(a) (b) (c)

Figure 11. Bias-variance trade-off Ridge, N=20, (a) $\lambda = 0.0001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$.



(a) (b) (c)

(d) (e) (f)

Figure 12. MSE train and test data Ridge, N=20, using Bootstrap: (a) $\lambda = 0.0001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$ and using Cross-Validation: (d) $\lambda = 0.0001$ (e) $\lambda = 0.01$ (f) $\lambda = 1$.

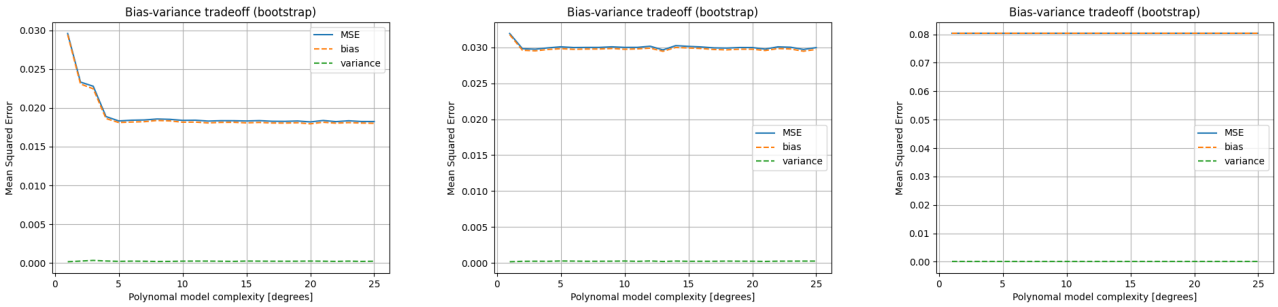


Figure 13. Bias-variance trade-off Lasso, N=20, (a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 0.1$.

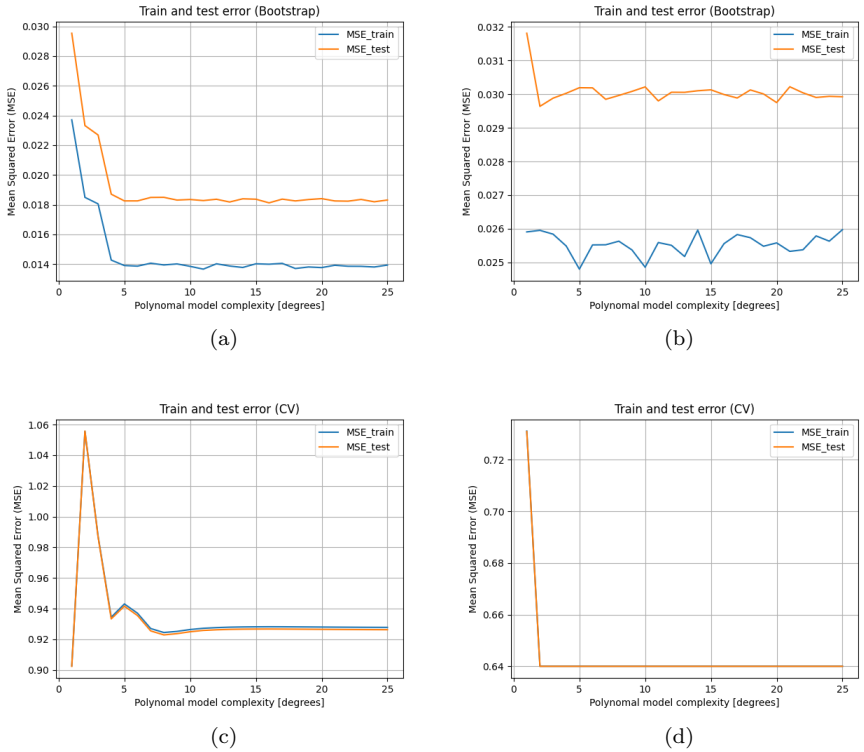


Figure 14. Bias-variance trade-off Lasso, N=20, using Bootstrap: (a) $\lambda = 0.001$ (b) $\lambda = 0.01$ and using Cross-Validation: (c) $\lambda = 0.0001$ (d) $\lambda = 0.01$.

E. Real Terrain Data

As a final test we will apply all of the prior algorithms on the real terrain data. All of the created figures can be found in the GitHub repository, so to avoid the clutter only two most relevant results will be included for the Ridge and Lasso algorithms.

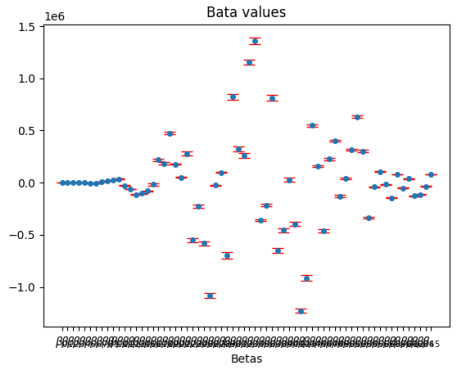


Figure 15. 95% confidence interval for β parameters, max_degree=30, N=20.

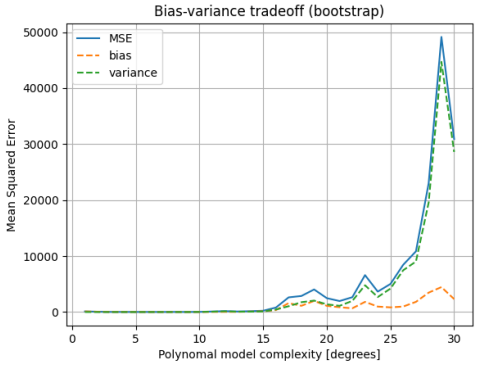


Figure 16. Bias-variance trade-off, OLS, N=20 for terrain data.

Following we have a comparison of the resampling method outputs where once again the k-fold cross-validation, here once again it feels like smaller values of λ need to be tested to get a proper overview over the performance of the model. Still we will take a look at λ values of $\lambda = 0.001$ and $\lambda = 0.01$ in Figure 22. In the case of the bootstrap algorithm we can see that the decrease of λ leads to improvement in the model, but the for cross-validation all we can realistically say that the values we were able to test do not provide a reasonable solution.

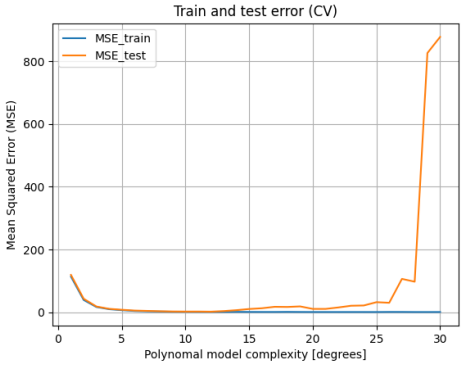
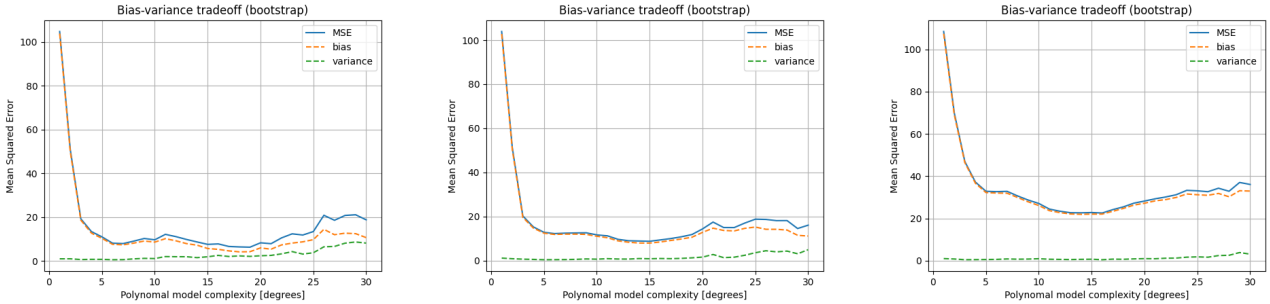


Figure 17. MSE train and test data OLS, Bootstrap, N=20 for terrain data.

Figure 18. MSE train and test data OLS, CV, N=20 for terrain data.

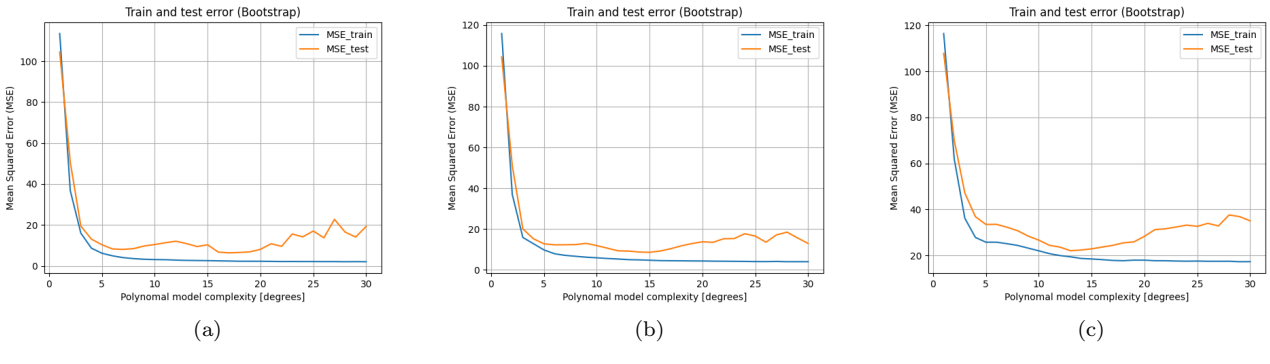
VI. CONCLUSION

We have employed three Linear Regression agorithms: OLS, Ridge and Lasso on the synthetic Franke function and on the real terrain data.While the algorithm each come with a certain list of upsides and downsides, OLS seems to have performed overall best over the course of this project, particularly for the lower model complexities. This might be caused by lack of the λ parameter. Performance at higher levels of complexity is improved using the Ridge regression algorithm. Further test will need to be employed to properly evaluate the Lasso algorithm, but as we stand its biggest downside is the computation time.

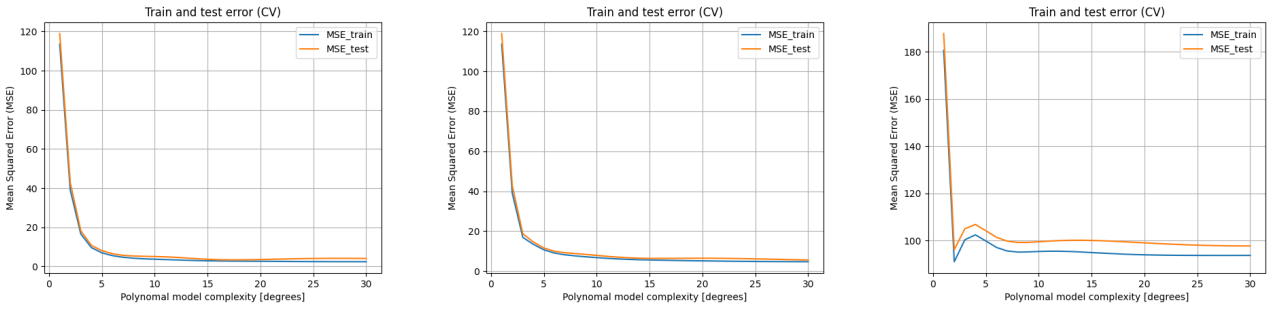


(a) (b) (c)

Figure 19. Bias-variance trade-off Ridge, N=20, for terrain data (a) $\lambda = 0.0001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$.

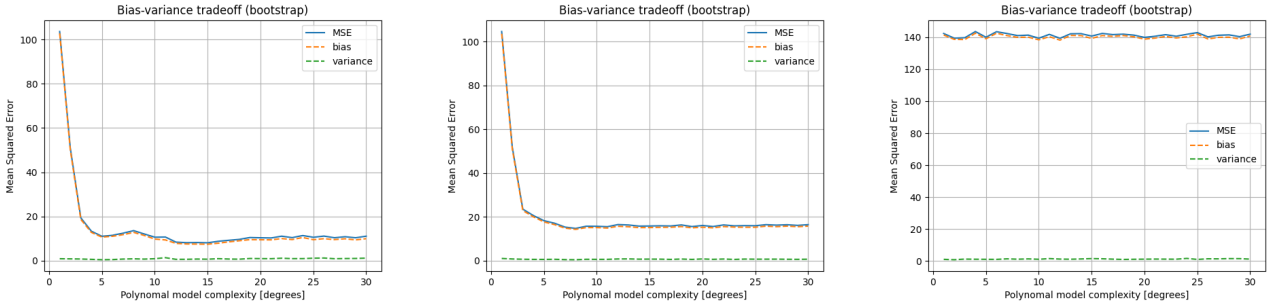


(a) (b) (c)



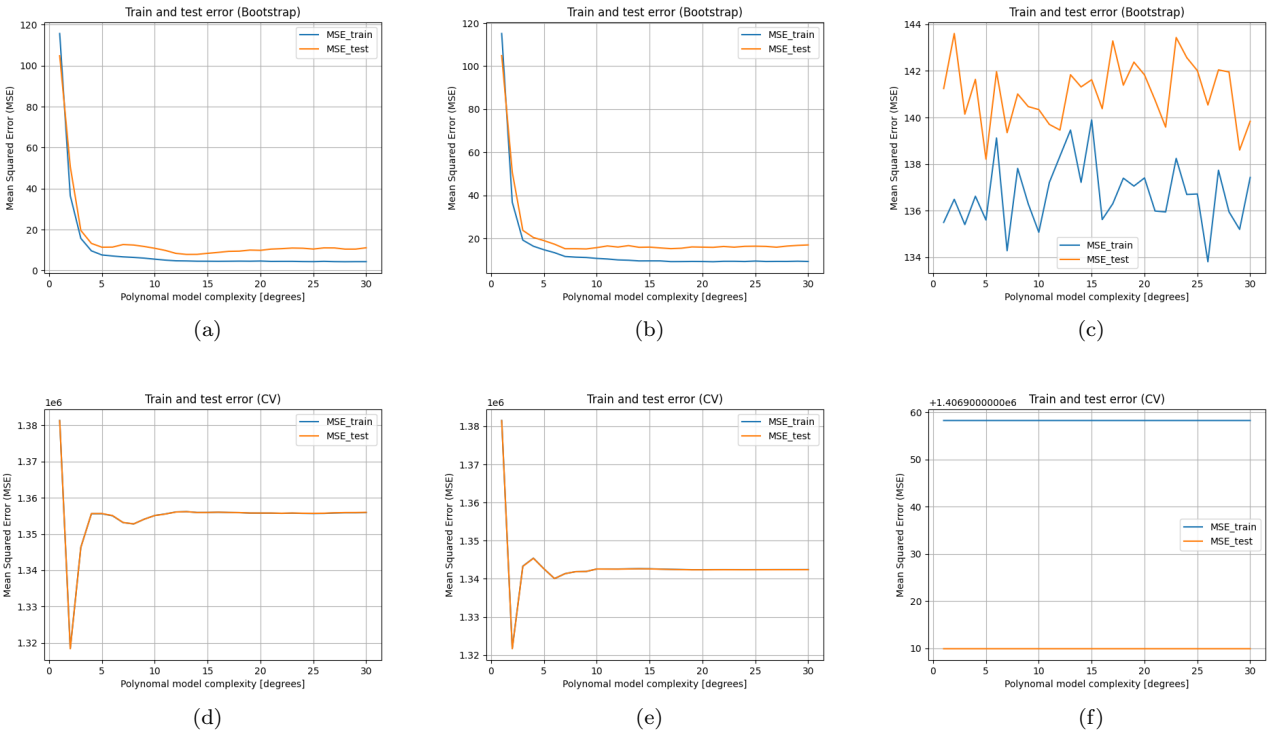
(d) (e) (f)

Figure 20. MSE train and test data Ridge, N=20, for terrain data, using Bootstrap: (a) $\lambda = 0.0001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$ and using Cross-Validation: (d) $\lambda = 0.0001$ (e) $\lambda = 0.01$ (f) $\lambda = 1$.



(a) (b) (c)

Figure 21. Bias-variance trade-off Lasso, N=20, (a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$.



(a) (b) (c)

(d) (e) (f)

Figure 22. Bias-variance trade-off Lasso, N=20, using Bootstrap: (a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 1$ and using Cross-Validation: (d) $\lambda = 0.0001$ (e) $\lambda = 0.01$ (d) $\lambda = 1$.

[1] Univeristy of Oslo, Department of Physics (2020) *Project 5 Diffusion Equation - description*, Available at: `http://compphysics.github.io/ComputationalPhysics/doc/Projects/2020/Project5/DiffusionEquation/pdf/DiffusionEquation.pdf`
[2] Franke, R. (1979). A critical comparison of some methods for interpolation of scattered data (No. NPS53-79-003). NAVAL POSTGRADUATE SCHOOL MONTEREY CA
[3] course GitHub https://github.com/CompPhysics/MachineLearning
[4] https://earthexplorer.usgs.gov/
[5] https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter1.html

APPENDIX