On Linear Regression Methods for a Terrain Dataset

Torstein Solheim Ølberg
Institute for Theoretical Astrophysics, University of Oslo
P.O. Box 1029 Blindern 0315 Oslo, Norway

October 12, 2021

Contents

1 Abstract **2**

2 Introduction **2**

3 Data Description **2**

4 Methods **2**
- 4.1 Scaling 2
- 4.2 Building the Model 3
  - 4.2.1 Ordinary Least Squares 3
  - 4.2.2 Ridge Regression 3
  - 4.2.3 Lasso Regression 3
- 4.3 Analysis methods 3
  - 4.3.1 Mean Square Error 3
  - 4.3.2 R Squared 4
  - 4.3.3 Bootstrap Resampling 4
  - 4.3.4 Cross Validation 4
- 4.4 Performing the analysis 4

5 Results **5**
- 5.1 Franke function 5
- 5.2 Terrain data 9

6 Discussion **13**

7 Conclusion **13**

Bibliography **13**

1 Abstract

Performing analysis in machine learning is a powerful tool in many different fields of science. For this it is important to familiarise oneself with different methods of analysis for different problems. In this project I look at three methods for approximating a 3D data set, Ordinary Least Squares, Ridge and Lasso regression. Use Mean Square Error, Bootstrap resampling and Cross Validation resampling to estimate which models fit a made up dataset, and a terrain dataset the best, and find that it is the 20th degree OLS fit which best mimic the datasets. As for Ridge and Lasso we find a $\lambda = -0.5$ and $p =$, and $\lambda = 0.1$ respectively, but the Lasso model doesn't converge for any polynomial degree and is thus useless.

2 Introduction

The field of data analysis has been revolutionised by the introduction of machine learning in the middle of the 20th century. This process revolves around the approximation of models to some set of data by the brute force use of computers. In this report we will try study three simple linear regression models, the Ordinary Least Squares (OLS), Ridge and Lasso regression methods, on a made up data set and then use it on a real dataset and try to find the best possible method. The article itself will start with a description of the made up data produced from the Franke function and the real data from the USA Geological survey. Then the scaling, model building and different analysis methods will be described. Finally the results are presented and discussed and the total contents of the article will be summarised.

3 Data Description

The first dataset we make ourselves, produced by the so called Franke function. This is a function consisting of the sum of four exponentials and produces a plane dataset since it takes two separate variables x and y.

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

We choose to make the dataset with x and y values from $[0,1]$, and choose linearly 10^4 pairs of points. We should also make the dataset more realistic by adding some noise to the data. For this we choose to draw a set of random numbers, equal to the number of data points, drawn from a normal distribution with mean value 0 and standard deviance equal to 1. Then we add these points to our Franke function values.

The second, and "real" dataset is taken from the United States Geological Survey [1]. Here we find an elevation dataset from the SRTM 1 Arc-second global group from the region around the Oslo Fjord. It is simply a measurement of the height in meters above sea level. We extract the small part in the top of the left corner showing Finnemarka. You can see the dataset in figure 1

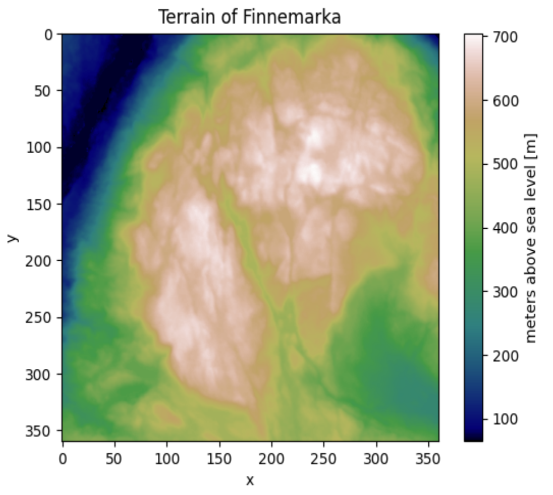

Figure 1: The Finnemarka elevation dataset

4 Methods

4.1 Scaling

It is useful to scale our data before we perform any approximation of our dataset, as this will help us to account for problems with outliers and particularly large values when we fit, which otherwise could make the data very dependant upon these points. We will use the simple standard scaling from *Sklearn.preprocessing* called scale. This subtracts the mean value of the data from the data points.

4.2 Building the Model

When performing a linear regression, we want a model on the form

$$z = \sum_{i=0}^{p} \sum_{j=0}^{i} \beta_{\sum_{k=0}^{i} + j} x^{i-j} y^j \tag{2}$$

This might look odd, but is just a compact way of writing a standard 2 dimensional p degree polynomial. for $p = 2$ this is simply

$$z = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 xy + \beta_5 y^2$$

To perform this regression we need to set up a vector $\vec{X}$ on the form

$$\vec{X} = \begin{bmatrix} \sum_{i=0}^{p} \sum_{j=0}^{i} x_0^{i-j} y_0^j \\ \sum_{i=0}^{p} \sum_{j=0}^{i} x_1^{i-j} y_1^j \\ \cdots \\ \sum_{i=0}^{p} \sum_{j=0}^{i} x_n^{i-j} y_n^j \end{bmatrix} \tag{3}$$

where n are the number of x, y pairs in the dataset. Then we can split our dataset into two groups, a training and a test group. We use 20% of our dataset as a test and the rest as training group.

4.2.1 Ordinary Least Squares

Now we are prepared to model our data with the OLS method. This is a simple task where we approximate the β values by finding the minimum value of the cost function

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

It is simple to show, by taking the derivative of C that the minimum point can be found from

$$\vec{\beta} = \left(\vec{X}^T \vec{X}\right)^{-1} \vec{X}^T \vec{y} \tag{4}$$

as long as $\vec{X}\vec{X}^T$ is invertible. If they are not, we can approximate them by the Moore-Penrose pseudo-inverse instead, and use this result. To compute the inverse we use the *numpy.linalg* function pinv [2].

4.2.2 Ridge Regression

Another way to perform the Linear regression is the Ridge method. This is almost the same as the OLS, except the introduction of a constant λ times the norm of β into the cost function.

$$C(\vec{\beta}) = \frac{1}{n} \left\{ \left(\vec{y} - \vec{X}\vec{\beta}\right)^T \left(\vec{y} - \vec{X}\vec{\beta}\right) \right\} + \lambda ||\beta||_2^2$$

This gives us a way to find beta in the same way as for the OLS except with a λ times an identity matrix

$$\vec{\beta} = \left(\vec{X}^T \vec{X} + \lambda \vec{I}\right)^{-1} \vec{X}^T \vec{y} \tag{5}$$

With the same assumption as for the OLS.

4.2.3 Lasso Regression

The last regression method is the Lasso regression, where we define the cost function to be equal to

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

where we have used $||\beta||_1 = \sum_i |\beta|$. This doesn't have a numerical solution to the derivative of C like the other two methods. However, the python package *sklearn* has a useful method *linear_model.Lasso* [3] which we can use to perform this minimalisation task for us.

4.3 Analysis methods

We need to find a way to measure which method is the best at approximating our dataset, and to do this a simple measure might be to look at the Mean Square Error and use this as our prediction error.

4.3.1 Mean Square Error

The MSE is simply the mean value of the square of the errors between the approximations and the measured values.

$$MSE = \frac{1}{n} \sum_i (z_i - \hat{z}_i)^2 \tag{6}$$

where z are the measured values and $\hat{z}$ are the true values.

4.3.2 R Squared

We can also calculate the R^2 value of the model, which is defined as

$$R^2 = 1 - \frac{\sum_i (z_i - \hat{z}_i)^2}{\sum_i (z_i - \bar{\hat{z}})^2} \tag{7}$$

$\bar{\hat{z}}$ is the mean value of the approximations.

4.3.3 Bootstrap Resampling

A more complicated, but still quite simple way to find the best fit model is to perform the bootstrap resampling method. This method is based on the assumption that we can perform the prediction of our data a number of times a, where we each time choose a random subset of the test and training group, with resampling of the training data. Thus we get a new train and test groups, with the same size as the original once, and we can find the MSE for each and averaging over all the MSEs. From this we can also split the MSE into three other parts, the Bias, the Variance and the variance of the error. To get this, we can look at the MSE

$$\begin{aligned} MSE =& \frac{1}{n}\sum_i (z_i - \hat{z}_i)^2 \\ =& \mathrm{E}\left[\left(\vec{z} - \hat{\vec{z}}\right)^2\right] \end{aligned}$$

where E denotes the expected value. If we add $E[\hat{\vec{z}}] - E[\hat{\vec{z}}]$ inside the square and perform the squaring, we can rewrite this as

$$\begin{aligned} MSE =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2\right. \\ &+ \left(\vec{z} - E[\hat{\vec{z}}]\right)^2 \\ &\left.+2\left(E[\hat{\vec{z}}] - \vec{z}\right)\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)\right] \end{aligned}$$

The estimated value of an estimated value is simply the estimated value, and the estimated value of a measured value is simply the measured value. This lets us rewrite the above as

$$\begin{aligned} MSE =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ E\left[\left(\vec{z} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ 2\left(E[\hat{\vec{z}}] - \vec{z}\right)\left(E[\hat{\vec{z}}] - E[\hat{\vec{z}}]\right) \\ =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ E\left[\left(\vec{z} - E[\hat{\vec{z}}]\right)^2\right] \\ =& \mathrm{Var}(\hat{\vec{z}}) + \sigma^2 + \mathrm{Bias}(\vec{z}, \hat{\vec{z}})^2 \end{aligned}$$

The Last step her is just splitting the left term into the variance of the model and the variance of the assumed noise. This result gives us the bias and the variance

$$\mathrm{Bias} = \frac{1}{n}\sum_i \left(z_i - E[\hat{\vec{z}}]\right)^2 \tag{8}$$

$$\mathrm{Var} = \frac{1}{n}\sum_i \left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2 \tag{9}$$

As we can see, the bias is simply the average of the difference between the measured value and the estimated approximation value. The Variance is the same, but the measured value is exchanged for the approximated value.
We can then calculate the MSE, bias and variance and plot them as a function of our polynomial degree, to find at what value the bias and variance has the optimal values.

4.3.4 Cross Validation

Last of all, we can perform the so called Cross Validation resampling method (CV). Here we shuffle and perform a random split of data into test and training groups a number of times N and calculate the MSE every time. Then we find the mean of this value, by summing all the results and dividing by N, and use this as our MSE estimate.

4.4 Performing the analysis

First we will perform the analysis methods for OLS, finding the best fit polynomial degree, which would be the polynomial degree giving the lowest error values. Then we will find the optimal lambda for Ridge and Lasso, using a degree of 5. This we do for the Franke function to test our method, afterwords we can perform the same thing for the terrain data. Last, we use this lambdas for the terrain data to find the optimal degree and compare the best fit models for OLS, Ridge and Lasso. Optimally we would have liked to perform the Lasso and Ridge regression for an interval of degree and λ at the same time, to be sure the combination we get is the best possible. This would however take much more time to run than we have time for so we will simply perform one at a time and assume this is close to the right result.

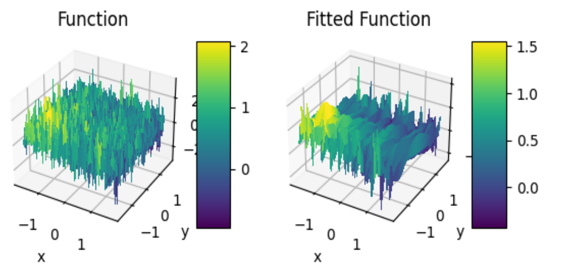

Figure 3: To the left is the Franke function with noise and to the right is the fitted approximation with OLS method for a fifth degree polynomial. They vary a lot from each other.

5 Results

5.1 Franke function

Performing the OLS estimate of the Franke dataset with a first through fifth degree polynomial, we get R^2 values as shown in figure 2. In this plot we see that the function increases in the beginning and then reaches a top, falls drastically and then rises a little again. This also gives us the visualisation of the approximation seen in figure 3 where the left plot is of the actual Franke function with noise added, and the right is the OLS approximation to the function for the fifth degree. We see that there are a lot of difference between the function with noise and the model.

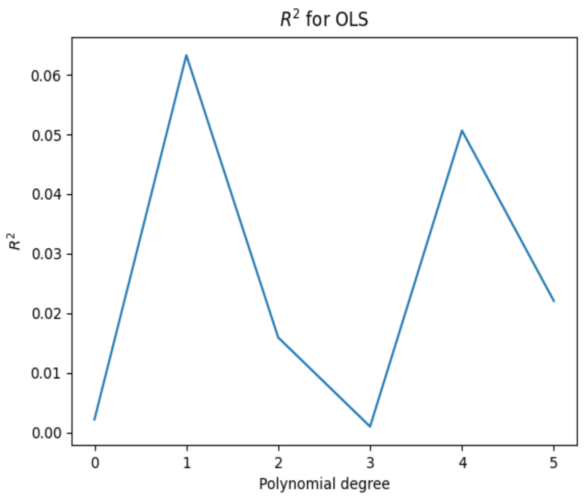

Figure 2: R^2 as a function of the polynomial degree for the OLS method applied on the Franke function. We see that the function rises until 3rd degree, then falls to a negative value for 4th degree and rises for the 5th one again.

When we perform the OLS for a range of polynomial degrees between zero and 30, we get a MSE as a function for the polynomial degree for both the training and test data. This plot is seen in figure 4.

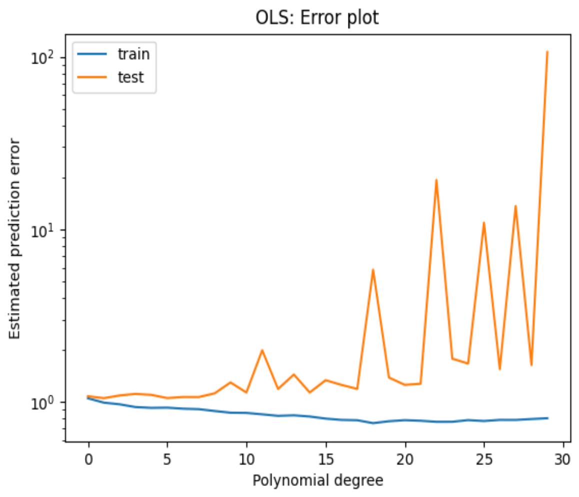

Figure 4: MSE as a function of polynomial degree for both the test and train dataset groups. Train data has its error reduced with higher degree, but the test data rises after 8th degree.

We see that the training data error is reduced the higher value of the polynomial we get, until around 20th degree where it stabilises. However the test data has a stable curve until around 8th degree

where it starts to vary drastically. This gives us a lowest error value at 7th degree for both sets.
The Bootstrap analysis for the OLS gives us a plot of the MSE, Bias and Variance, which we can see in figure 5.

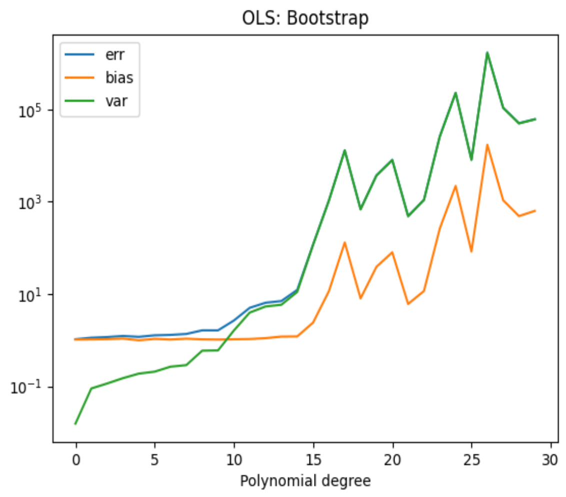

Figure 5: Bootstrap MSE, bias and variance for OLS. Error is stable until around 7th degree and then blows up, bias is stable until 14th degree and then blows up and variance increases slowly until 14th degree when it overlaps with error and blows up.

Here we see a similar evolution to the test curve in the MSE plot. The error and bias stay almost constant until about 8th level, where it stats to increase and vary drastically. The exception is the variance which increases slowly the whole time but blows up after 10th degree.
The Cross Validation gives us a MSE which we can compare with the MSE of Bootstrap. We plot these two in figure 6.

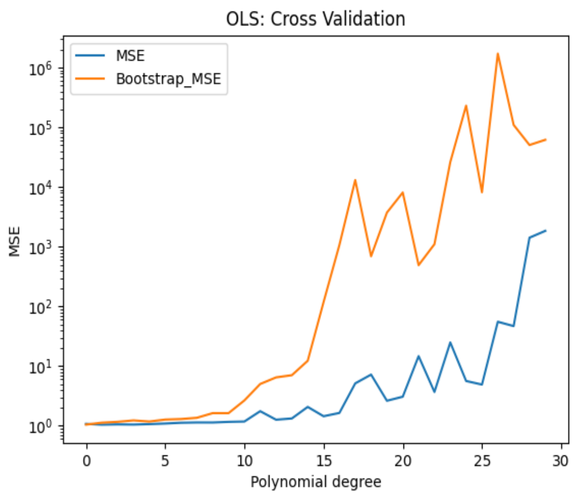

Figure 6: Cross Validation MSE estimate for OLS compared with the Bootstrap MSE. Both are very similar and almost constant until around 7th degree where the bootstrap start to blow up. Then later at 10th degree the CV also start to increase. The bootstrap error increases much faster than the CV MSE.

We see that the CV is stable like the bootstrap was, but start to increase later than the bootstrap, waiting until around 1st degree before it increases more slowly but blows up eventually.
For the Ridge we get a MSE as a function for the λ for both the training and test data which we can see in figure 7.

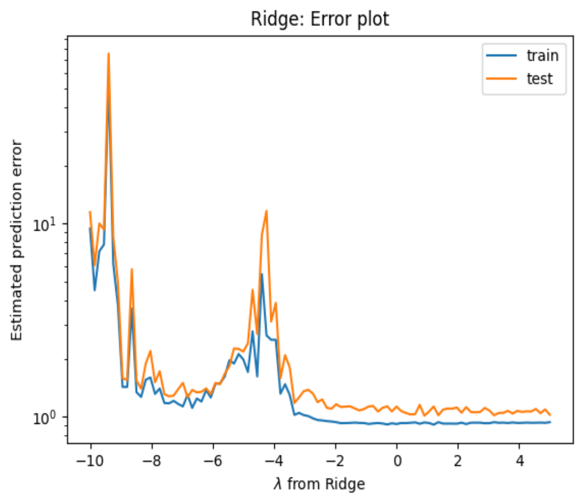

Figure 7: MSE as a function of lambda for both the test and train data set groups of the Ridge model for the Franke function. We see a local minimum for the negative values of λ at around negative seven, but at negative 1 the error reduces to around 1 and stabilises at that value after pasing $\lambda = 0$. This evolution is almist identical for both the train and test datasets.

We see that both dataset groups MSEs look mostly the same. They fluctuate a little for the negative values with a local minumum at $\lambda = -7$ and reach a minimum around zero where they stabilise. The Bootstrap for the Ridge gives us a plot of the MSE, Bias and Variance as seen in figure 8.

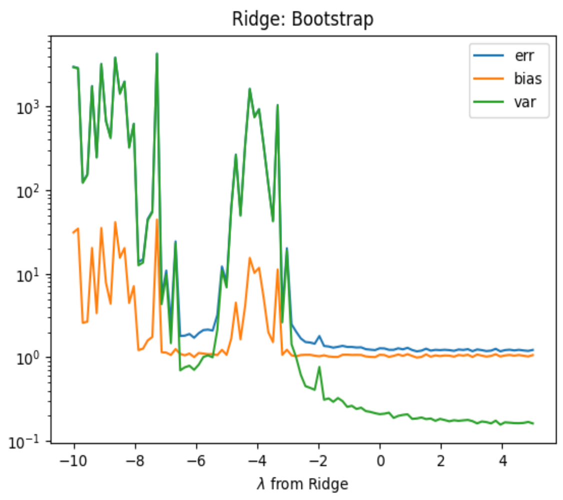

Figure 8: Bootstrap MSE, bias and variance for Ridge model of the Franke function. The three curves fluctuate erratically until $\lambda = -1$ where they stabilise at a minimum value. the exception is variance which continue to decrease.

We see all three quantities fluctuate for negative values of lambda, but stabilise after negative 1 for quite a low value except variance which continue to decrease.
The CV gives us a MSE which we can compare with the MSE of Bootstrap as we have done for OLS. We have plotted this in figure 9.

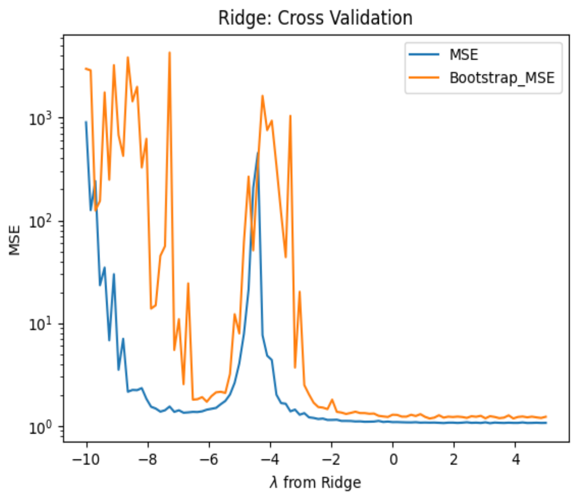

Figure 9: CV MSE estimate for Ridge compared with the Bootstrap MSE for the Franke function. Both have a local minimum at negative six and stabilise at a minimum after minus one.

We see that this the two curves are almost identical with a local minimum at minus six and a stable minimum after reaching minus one.
For the Lasso we get a MSE as a function for the lambda value for both the training and test data. this is plotted in figure 10.

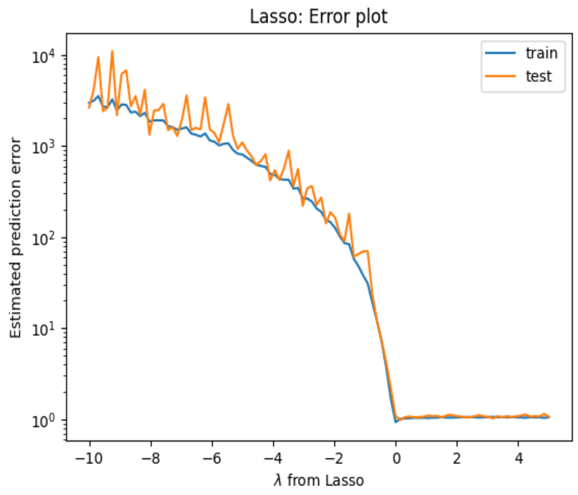

Figure 10: MSE as a function of lambda for both the test and train dataset groups of the Lasso model with the Franke function. We see a decreasing curve until $\lambda = 0$ where the error stabilises at a minimum.

We see that both dataset groups have the same evolution, falling down at zero and staying the same after this.
The Bootstrap for the Lasso gives us a plot of the MSE, Bias and Variance given in figure 11.

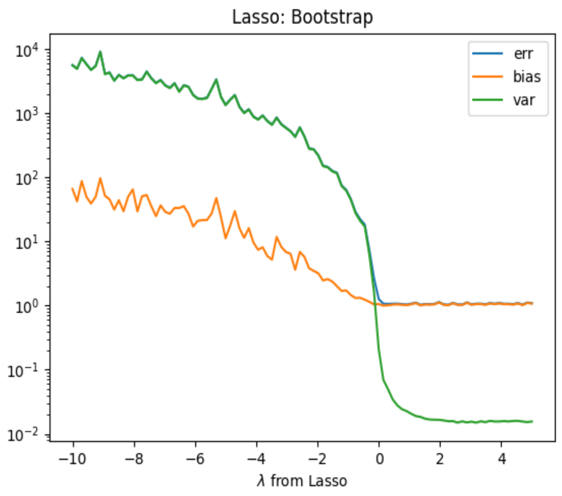

Figure 11: Bootstrap MSE, bias and variance for Lasso. Has the same general form as the error plot, but we can't see the error because it follows the varians values so much at negative λ and bias at positive.

Varians starts high but drop off a lot at zero. Until then, the error follows this. Then, at lambda equal to zero, the bias passes the variance and the error starts following this.
The CV of the Lasso model of the Frake function gives us a MSE which we can compare with the MSE of Bootstrap like earlier, which we see in figure 12.

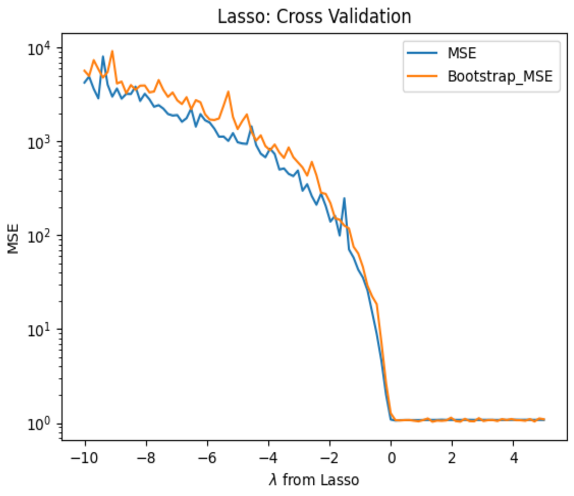

Figure 12: CV MSE estimate for Lasso compared with the Bootstrap MSE for the Franke function. The CV and Bootstrap look almost identical, falling until zero where they flatten out.

We see that the CV and Bootstrap gives us the same form. They fall of until $\lambda = 0$ where they flatten out.

5.2 Terrain data

Then, with the terrain data, we get an OLS, Ridge and Lasso model. For the OLS model we get three analysis plots, using the MSE, Bootstrap and CV methods. We can see these plots in figure 13, 14 and 15.

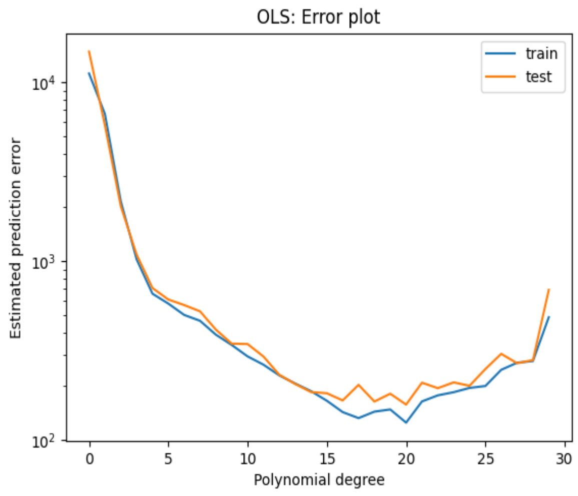

Figure 13: MSE for OLS for both training and test groups with the terrain dataset. They decrease to a minimum at around 20th degree befor they start to increase again.

In figure 13 we see that the terrain data decreases to a minimum around 20th degree befor starting to rise again, for both training and test dataset.

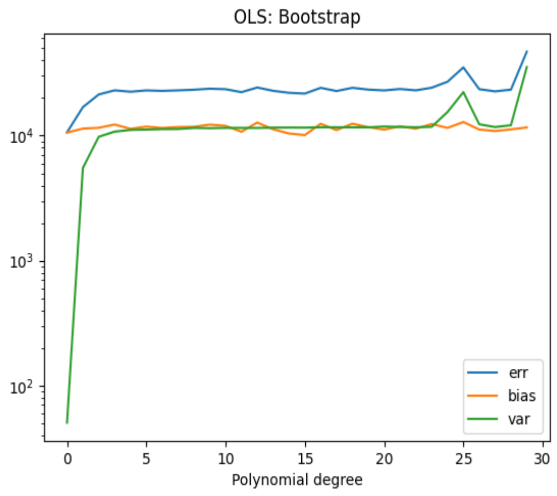

Figure 14: MSE, Bias and Variance for OLS with the terrain dataset. Bias is just constant with very minor fluctuations for all the polynomial degrees, while error and variance increase in the beginning, flattens out and stays flat until both have a spike at around 25 and then start rising just before 30.

The Bootstrap analysis in 14 show an almost constant flat line for al variable, with an increase in value at the start and a spike at around 25th degree and a rising curve just before 30th.

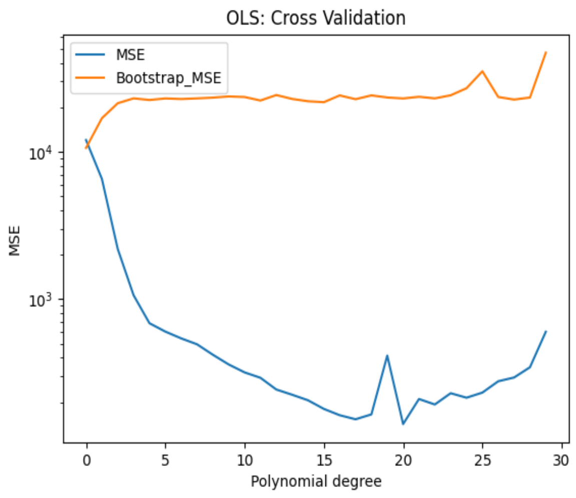

Figure 15: MSE for OLS for both CV and Bootstrap with the terrain dataset. Here we see a minimum for around 20th degree with a fall off in value before and a rise in value afterwords.

The Cross Validation plot in figure 15 show the same trend as the MSE did, with a minimum at

polynomial degree of 20. This is a totally different evolution to the MSE from the Bootstrap.
We see that two of the three plots gives us a minimum at around 20th degree. The Bootstrap result is almost constant for all polynomial degrees.
For the Ridge analysis of the terrain data, we also get three analysis plots. These are shown in figure 19, 17 and 18.

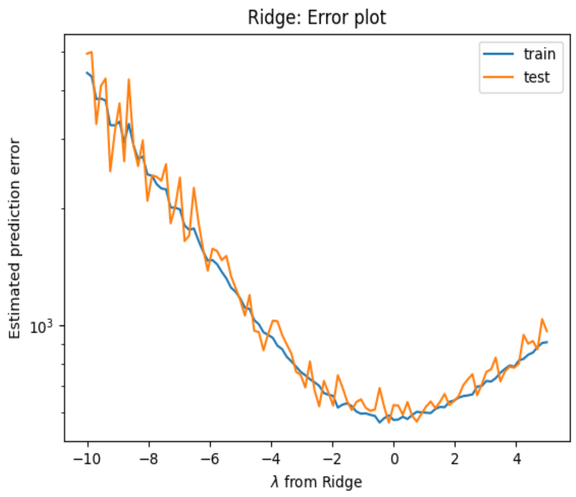

Figure 16: MSE for Ridge for both training and test groups with the terrain dataset varying with different values of lambda. We see a general decreasing curve until around zero, with a minimum just before at possibly minus a half.

For the MSE plot in figure 13 we see a trend where both dataset fall and reach a local minimum at around zero with a possible minimum at minus a half, before rising upwards again.

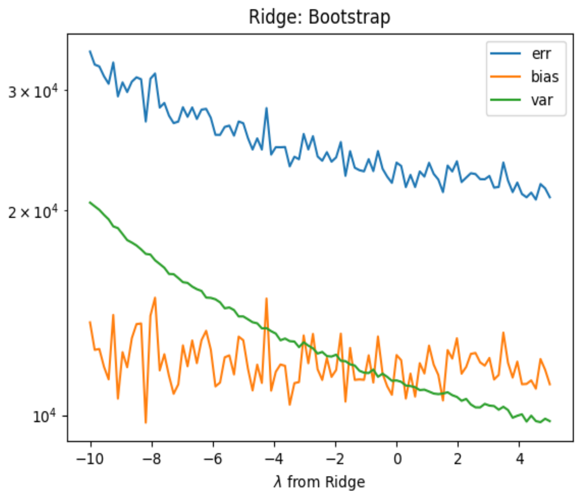

Figure 17: MSE, Bias and Variance for Ridge with the terrain dataset, for varying lambdas. Here we see that the error and variance decrease slowly for higher values, while the bias just fluctuates around a constant value.

In figure 17 we see the Bootstrap analysis of the Ridge model for different values of lambda. Here the error and the variance drop slowly for higher values of λ, while the bias fluctuate around the same constant value.

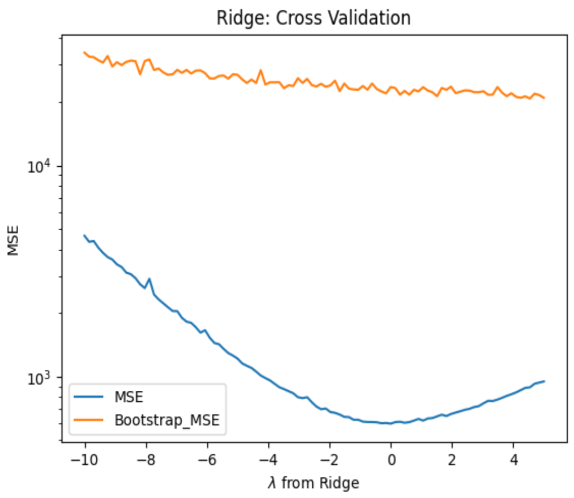

Figure 18: MSE for Ridge for both CV and Bootstrap with the terrain dataset. Here we see that the CV MSE has a minimum just before zero, while the Bootstrap analysis just falls off for higher values. The general error of the Bootstrap method is also much higher than for the CV.

The Cross Validation analysis shown in figure 18

decrease and has a minimum just like the MSE had, unlike the Bootstrap error.
We see that the MSE and CV gives us a minimum lambda value, but the Bootstraps just drops of for rising values.
For the Lasso we get figure 19, 20 and 21.

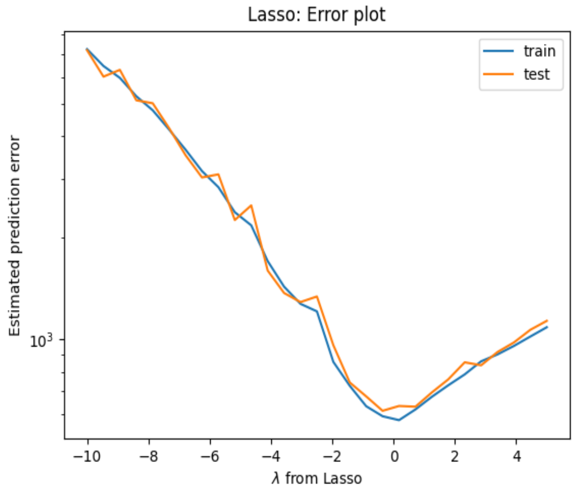

Figure 19: MSE for Lasso for both training and test groups with the terrain dataset. Here the error decrease sharply until a minimum at about $\lambda = 0.1$ before rising a little slower up again.

In the MSE analysis in figure 19 we see a sharp decrease until a minimum at around 0.1 when the error start to increase again. The two datasets evolve almost identically.

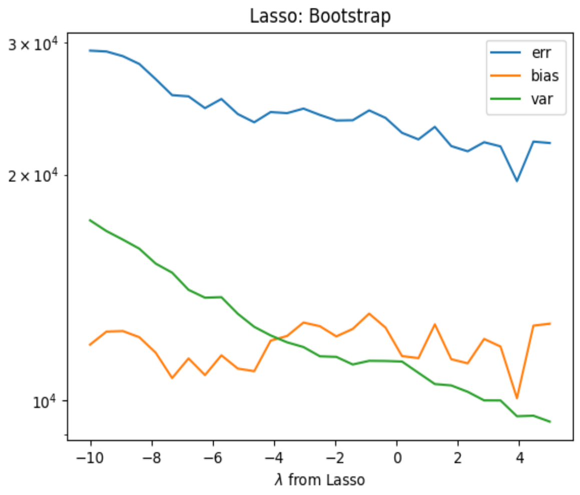

Figure 20: MSE, Bias and Variance for Lasso with the terrain dataset. The error and variance drop slowly for higher lambda, but the bias varies a around some value. The bias and the error do however seem to increase and decrease at the same lambda values, but with different amounts, giving them a very similar shape.

In the Bootstrap analysis in figure 20 we see the error and variance decrease for higher values of lambda while the bias just varies around a value.

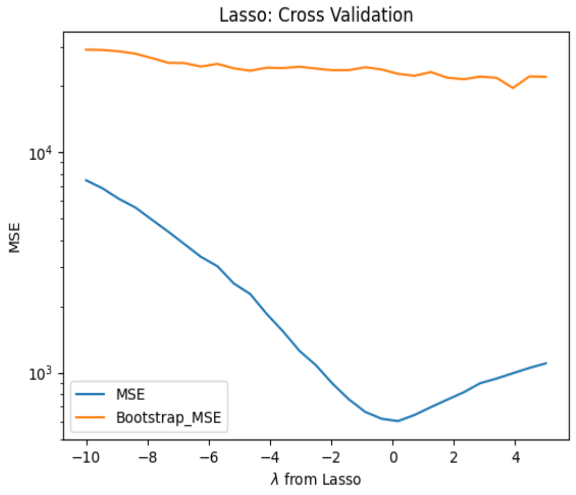

Figure 21: MSE for Lasso for both CV and Bootstrap with the terrain dataset. The CV MSE drops to a minimum just after zero, unlike the Bootstrap MSE which decreases slowly with no end.

Last of all we see that in figure 21 we get the same shape as the MSE curves, with a minimum

just after zero, unlike the Bootstrap which just decreases.
We see that the error and CV gives us a minimum lambda value, but the Bootstraps just drops of for rising values just like we saw for Ridge.
Using these optimal values, we get the three analysis plots for Ridge as functions of polynomial degree. These can be seen in figure 22, 23 and 24.

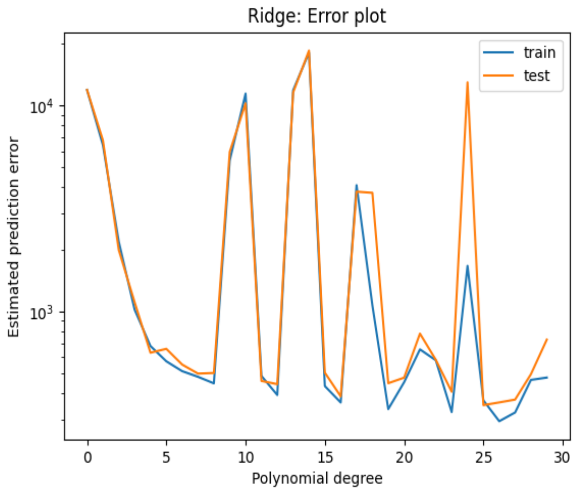

Figure 22: MSE for Ridge for both training and test groups with the terrain data set as a function of polynomial degree with $\lambda = -0.5$. The MSE decrease sharply until it reaches 8 when it starts to vary up and down, very fast, between maximum and minimum values.

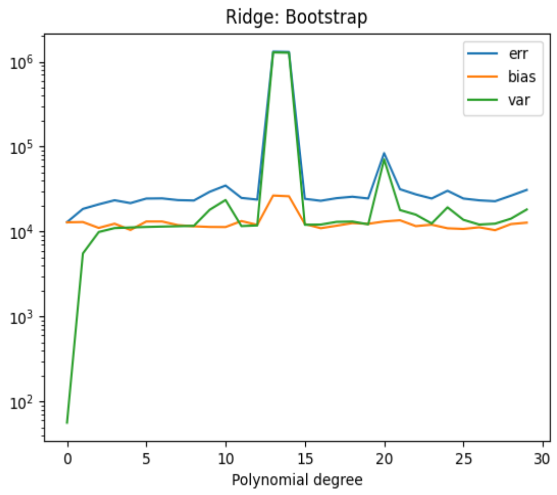

Figure 23: MSE, Bias and Variance for Ridge with the terrain data set as a function of polynomial degree with $\lambda = -0.5$. Here we see that the error, bias and variance fluctuate a little around a constant value, except for a few spikes in the error and variance where the most notable are at 13th and 20th degree.

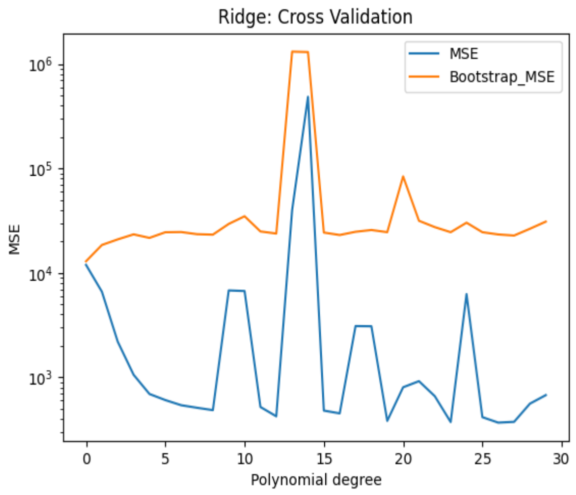

Figure 24: MSE for Ridge for both CV and Bootstrap with the terrain data set as a function of polynomial degree with $\lambda = -0.5$. The CV MSE has about the same evolution as the MSE, which for one is not that far off from the Bootstrap except for the much larger values.

We see that the error and CV gives us a local minimum polynomial degree at around 8 before it starts to vary a lot, but the Bootstraps stay constant for most of the plot, with only two large spikes

at 13th and 20th degree.
For the optimal models we get the following estimated errors

	OLS	Ridge
MSE	124.4128106	439.1661682
Bootstrap MSE	22536.47117	24029.23005
CV MSE	155.1733531	473.8888207

Figure 25: Estimated errors for the two methods, OLS and Ridge with the three different analysis methods. The OLS gives overall better values for all estimates. The Bootstrap values are much higher than the MSE and CV values are.

6 Discussion

From the analysis of the Franke function we can see that the OLS method doesn't seem to give a best fit value, but it indicates that the best models are all those before 7th polynomial degree. As for the Ridge and Lasso, They both indicate a value very close to zero for the λ though they also could be any values above zero as well.The Bootstrap results are odd for every regression method. Though the mean square error looks like it is the sum of the variance and the bias like we would expect, the variance follows the same evolution of the bias, instead of being the opposite like it should. This indicates that there might be a dependence for the variance upon the bias, which there shouldn't be, and that this analysis method here is not as reliable as the other two, It also gives a much higher error than the other two methods, indicating something might be wired, though this might be just a result of it being a different error method. As for the Cross Validation we see about the same evolution as the Error. As for the terrain data, we see almost the same things as for the Franke function, but we can still use the MSE and the CV to find an optimal lambda and polynomial degree. We find $p = 20$ as the optimal for OLS, $\lambda = -0.5$ and $p = 8$ for the Ridge and $\lambda = 0.1$ for the Lasso. However, for the Lasso regression, with the best fit lambda we couldn't get the Lasso regression to converge into a best fit, making this method unavailable. From the calculations of the errors of these best fits, we have that the OLS gives the best fit for our dataset for all the error estimation methods.

A problem we have overlooked in this analysis is that we have not calculated confidence interval. This would have been a simple task, just calculating the MSE of the standard deviance if we assumed some distribution model, but we didn't have the time to do so unfortunately.
Another problem is that we have chosen to perform the estimation of lambda first, and then the estimation of polynomial degree for the Ridge. This was done to save time, as the calculations where very time consuming. This means we assumes that the two parameters λ and p are unrelated, which is not an assumption it is possible to choose without testing, since the β variable is a function of λ and the higher the polynomial degree, the more betas are in the model.

7 Conclusion

We saw that all the methods estimate a best fit for the Franke function, and for the terrain data, except that the Lasso doesn't converge for the polynomial degree. However we get a best fit for the other two, and from our best fit models we end up with the model best fitting our terrain data is the Ordinary Least Squares model of a 20th degree polynomial. For future work, it would be useful to find a way to reduce the time consuming part of the code, which seems to be the Lasso and Ridge λ analysis such that these can be performed at the same as the polynomial analysis. This would give a more robust model. It would also be useful to look into the Bootstrap method more closely and find out why it behaves so odd compared to the others.

References

[1] United States Geological Survey;
Earth explorer
`https://earthexplorer.usgs.gov/`
Published 23/09-2014
Downloaded 10/10-2021

[2] The NumPy community;
numpy.linalg.pinv
`https://numpy.org/doc/stable/reference/generated/numpy.linalg.pinv.html`
Published 22/06-2021
Downloaded 10/10-2021

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.; Scikit-learn: Machine Learning in Python, volum 12, page 2825-2830
Journal of Machine Learning Research, 2011
https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html