FYS-STK4155: Project 1

Simen Nyhus Bastnes

5. October 2020

Abstract

In this project, we will study various methods of linear regression, namely the Ordinary Least Squares method, Ridge, and Lasso regression, as well as investigate resampling the data via Bootstrapping and k-fold Cross-Validation. The data we will be looking at is the so-called Franke function, as well as terrain elevation data from a region in Norway. The Franke function gives us a way of testing our implementation before moving on to the more complex terrain data.

1 Introduction

With the emergence of more powerful computers, the field of machine learning is steadily becoming an integral part of both business and many fields of science. While many of the concepts and algorithms used in machine learning today has been known for a long time, some of them have simply been too computationally expensive to do efficiently. While not typically too computationally heavy **do something about this**, linear regression is one of the simplest and most-studied forms of machine learning, and provides a good introduction to concepts commonly used in machine learning.

In this project, we will look at three different methods of regression analysis and compare how they fare against each other. The methods we will be using is the Ordinary Least Squares method, Ridge regression, and Lasso regression. We will also see how resampling the data affects the results from the regression methods, by implementing the Bootstrap algorithm and the k-fold Cross-Validation.

There are two different data sets that will be studied in this project. The first, is the Franke function from [1], as well as terrain data for a region in Norway taken from [2]. First, in Chapter 2 we will introduce the theory behind linear regression, as well as the the regression methods and resampling methods employed later in the project. In Chapter 3 we go through the implementation of the methods, explaining how the code is structured and used. Then, in Chapter 4 we go through the results of both the Franke function and the terrain data, while discussing them. Lastly, we conclude our findings in Chapter 5.

2 Theory

2.1 Linear regression

Linear regression is a method of fitting a set of p *predictors* $\boldsymbol{X}$ to a data set $\boldsymbol{y}$, while minimizing the error between the *response* $\tilde{\boldsymbol{y}}$ and the actual data $\boldsymbol{y}$. For each of the n samples y_i in the data set the relationship between the response and the predictors $\boldsymbol{X}_i$ is modeled in a linear fashion, giving us the following matrix equation

$$\mathbf{y} = \mathbf{X}\beta + \boldsymbol{\epsilon}$$

where $\beta = (\beta_0, \beta_1, ..., \beta_{p-1})^\mathsf{T}$ are the regression parameters we are trying to estimate, one for each predictor, and $\boldsymbol{\epsilon}$ is the error in our approximation. The matrix $\mathbf{X}$ is often called the design matrix, and the equation can be written a bit more explicitly as

$$y_i = \beta_0 + X_{i,1}\beta_1 + ... + X_{i,p-1}\beta_{p-1} + \epsilon_i$$

Exactly what each predictor is can vary a lot from case to case, and how the design matrix is set up is important for the accuracy of the fit. In our case, we will focus on a form of linear regression where the predictors is on the form of a polynomial in the input parameters. In the case where we have a data set $\boldsymbol{y}(\boldsymbol{x})$, the design matrix can for example be written on the form of

$$\mathbf{X} = (\boldsymbol{x}^0, \boldsymbol{x}^1, ..., \boldsymbol{x}^{p-1})$$

With that said, we still need some way to find the β's that fit the data best, and we will now look at three ways to try to do this.

2.1.1 Ordinary least squares

Following Chapter 2.3 of Hastie et al. [3], in order to find the optimal regression parameters β, the OLS method minimizes the residual sum of squares

$$\text{RSS}(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2$$

With $\boldsymbol{y}$ as the vector containing all N y_i, and X an $N \times p$ matrix as shown in section 2.1, this can be written as

$$\text{RSS}(\lambda) = (\boldsymbol{y} - \mathbf{X}\beta)^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta)$$

Differentiating with respect to β we get

$$\frac{\partial \text{RSS}}{\partial \beta} = X^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta)$$

In order to find an optimal β, this has to be zero

$$\begin{aligned} \mathbf{X}^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta) &= 0 \\ \mathbf{X}^\mathsf{T}\mathbf{X}\beta &= \mathbf{X}^T\boldsymbol{y} \end{aligned}$$

Finally giving us the expression for the optimal regression parameters

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

assuming that $\mathbf{X}^\mathsf{T}\mathbf{X}$ is invertible.

2.1.2 Ridge regression

Ridge regression is an example of a so-called shrinkage method, which shrinks the regression coefficients by adding a small penalty proportional to their size.

$$\beta^{\text{Ridge}} = \min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_2^2$$

where λ is a regularization parameter that controls the amount of shrinkage, and we $||\beta||_2^2 \leq t$ where t is a finite number larger than zero. The higher λ, the more shrinkage occurs. This can be shown to give the Ridge solution

$$\beta^{\text{Ridge}} = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda I)^{-1}\mathbf{X}^\mathsf{T}\boldsymbol{y}$$

The aim with Ridge regression is to limit the potential problems with singularities when computing the inverse of $\mathbf{X}^\mathsf{T}\mathbf{X}$, which can be a problem when there are many correlated variables.

2.1.3 Lasso regression

Lasso regression is another shrinkage method, with a slightly different optimization equation compared to Ridge regression

$$\beta^{\text{Lasso}} = \min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_1$$

where $||\beta||_1$ is the L_1 norm.

2.2 Bias-Variance decomposition

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

2.3 R2????

2.4 Confidence intervals

For the OLS and Ridge regression cases, it is possible to derive the variance of β (a proper derivation is given in [4]), and thus the confidence intervals as well. For the OLS method, the variance is given by

$$\text{Var}(\beta) = \sigma^2 (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$$

where σ^2 is the estimated variance of y given by

$$\sigma^2 = \frac{1}{N - p - 1} \sigma_{i=1}^{N} (y_i - \tilde{y}_i)^2$$

Taking the square root of the diagonal of $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$ gives us an estimate of the variance of the j-th regression coefficient

$$\sigma^2(\beta_j) = \sigma^2 \sqrt{[\mathbf{X}^\mathsf{T}\mathbf{X}]_{jj}^{-1}}$$

Letting us construct the 95% confidence intervals by

$$\mathrm{CI}(\beta_j) = \left[\beta_j - 2\sqrt{\sigma_2(\beta_j)}\,,\;\; \beta_j + 2\sqrt{\sigma_2(\beta_j)}\right]$$

Similarly, the variance for β for Ridge regression can be found to be

$$\mathrm{Var}[\beta^{\mathrm{Ridge}}] = \sigma^2[\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^\intercal\mathbf{X}[(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}]^\intercal$$

and confidence interval can be constructed following the same steps as done above for OLS.

2.5 Resampling methods

In order to assess our models properly, we will be using some form of resampling. Specifically, we will be looking at the Bootstrap, and the standard k-fold Cross-validation resampling methods.

2.5.1 Bootstrap

The basic concept of the Bootstrap resampling method is to create new data sets by drawing samples (with replacement) from the training data set. Algorithm 1 shows the Bootstrap method given a data set $\mathcal{L}$ consisting of the data $\mathbf{X}_\mathcal{L} = \{(y_j, \boldsymbol{x}_j), j = 0 \dots n-1\}$

Algorithm 1 Bootstrap

```
1: Split the data set $\mathbf{X}_\mathcal{L}$ into training $\mathbf{X}_{\mathcal{L},\mathrm{train}}$ and test data sets $\mathbf{X}_{\mathcal{L},\mathrm{test}}$
2: for $i = 0,\ N_{\mathrm{bs}} - 1$ do
3:     Create a new data set $\mathbf{X}_{\mathcal{L},i}$ by drawing samples with replacement from $\mathbf{X}_{\mathcal{L},\mathrm{train}}$.
4:     Fit the model using $\mathbf{X}_{\mathcal{L},i}$.
5:     Evaluate the model on the test set $\mathbf{X}_{\mathcal{L},\mathrm{test}}$ and store the results.
6: end for
7: Assess the model by looking at the distribution of computed quantities, for example looking
   at the mean of the MSE.
```

2.5.2 Cross-validation

Cross-validation is a resampling method where (in the case of k-fold CV) the data set is split into k-folds of training and test data sets. Algorithm 2 shows the standard k-fold cross-validation.

Algorithm 2 k-fold Cross-Validation

```
1: Shuffle the data set $\mathbf{X}_\mathcal{L}$ randomly.
2: Split the data set $\mathbf{X}_\mathcal{L}$ into $k$ folds/subsets $\{\mathbf{X}_{\mathcal{L},i}, i = 0 \dots k-1\}$.
3: for $i = 0,\ k - 1$ do
4:     Set $\mathbf{X}_{\mathcal{L},i}$ as the test data set and the rest of the folds as the training set.
5:     Fit the model using the training data set as defined above.
6:     Evaluate the model on the $i$-th test set $\mathbf{X}_{\mathcal{L},i}$ and store the results.
7: end for
8: Assess the model by looking at the distribution of computed quantities.
```

3 Implementation

maybe put data sets after code?

3.1 Code

need to explain how noise is normal distributed 1. Write about what has been implemented 2. Slightly explain structure The heart. [5]

maybe conclusion? WE REWROTE THE CODE ON ROCK AND ROLL The code as it stands is divided into sections based on the different tasks in the project. A better approach, would probably have been to combine all the regression and resampling into a single code section, and rather have the run modes determining which parameters and regression methods are used and what is plotted. This is essentially how the section for the terrain data currently is operating. This would help a lot with reducing the total amount of code needed, and also making it easier to fix issues since there is only one section of the code that is computing the regression, and not multiple.

3.2 Data sets

In this project, we will be using two different data sets. The first, is the Franke function shown in appendix A.1. This function is a two-dimensional function that has been widely used for testing implementations for regression and interpolation.

The second data set is terrain elevation maps taken from [2], and the maps are stored in the GeoTIFF file format, and are from the SRTM Arc-Second Global data set. In the `datafiles` folder there are three maps, each of one region of Norway, though only one of them, `SRTM_data_Norway_3.tif` is studied in this project. This map is over the south-eastern part of Norway. A plot of the entire map, as well as a more detailed description of the location is given in figure 7 in appendix A.2.

4 Results and discussion

In this section, we will be showing some select results from running the code described in Chapter 3. More figures and data files, as well as some simpler benchmark runs can be found within their respective folders in the Github repository [5]. The results will be split into two sections, the first one pertaining to the Franke function, while the second is the analysis of the terrain data.

4.1 Franke function

First of all, we plot the Franke function with randomly drawn x, y samples, without noise, which can be seen in figure 1.

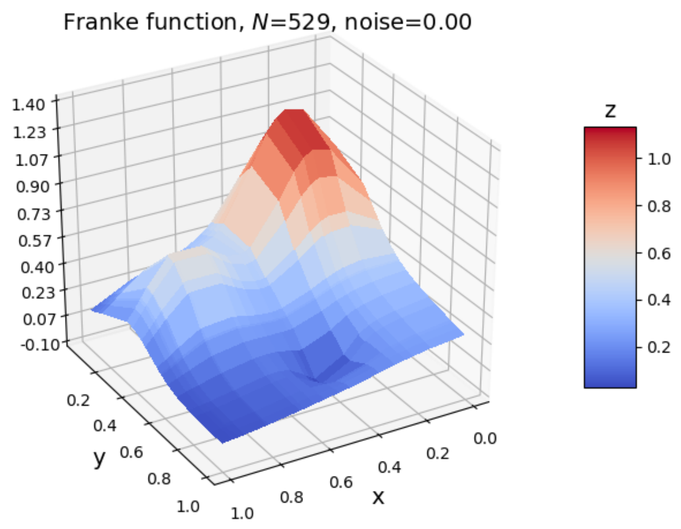


Figure 1: Example of the Franke function with no noise, with 529 randomly drawn points where $x, y \in [0, 1]$.

We start the regression analysis by performing the OLS method on the data set. Table 1 shows the MSE and R2 score for the train and test set where $p = 5$, $N = 529$. For this polynomial degree, both the training and test set shows good results, indicating that there is no significant overfitting.

Table 1: Mean squared error and R2 score for the Franke function training and test set when $p = 5$, $N = 529$, and noise $\sigma = 0.05$

Data set	MSE	R2
train	0.003334	0.949904
test	0.004674	0.929855

The 95% confidence intervals for β_{OLS} is shown in figure 2, with the same parameters. The confidence intervals are quite close to the coefficients, which makes sense given the MSE and R2 scores we found earlier.

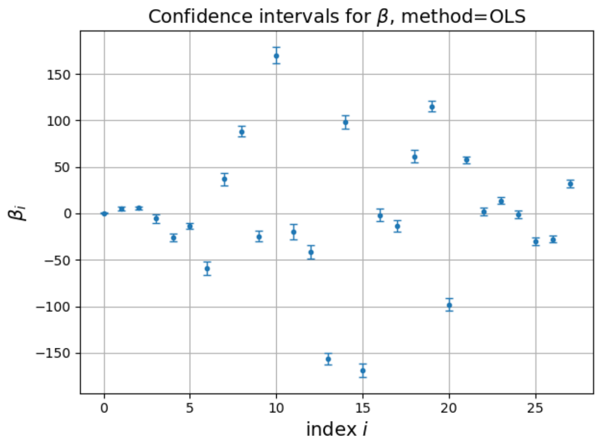
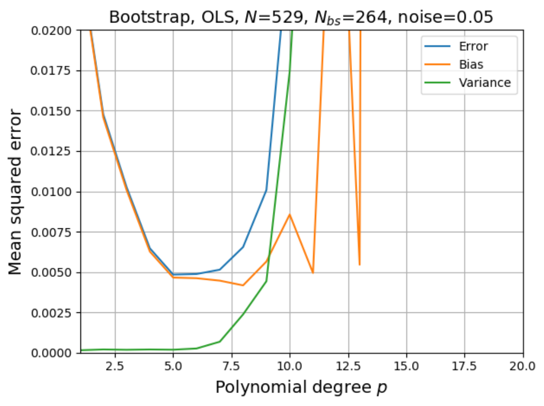
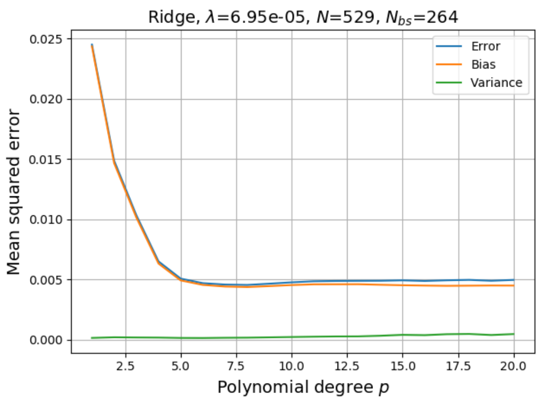


Figure 2: 95% confidence intervals for the regression coefficients β_{OLS} for $p = 6$, $N = 529$, and noise $\sigma = 0.05$.

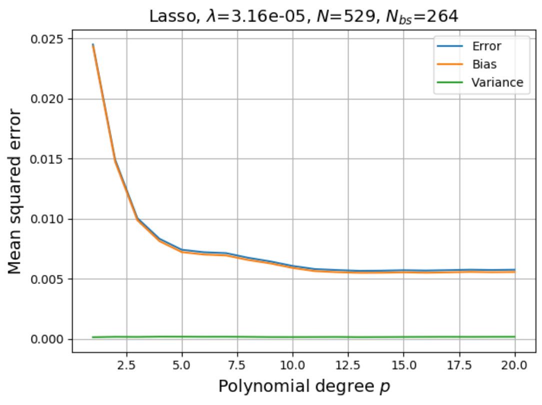
To study how the model is affected by the degree p, we study the bias-variance trade-off by employing the Bootstrap resampling method. Now, we will be studying all three regression methods, OLS, Ridge and Lasso to make comparing easier. Figure 3 shows the bias-variance trade-off for all three methods, with the Ridge and Lasso ones corresponding to the λ that gave the lowest MSE. We see that for the OLS, once we go beyond $p = 7$, the error and variance skyrockets. For Ridge and Lasso however, everything is fairly stable for $p > 5$. This shows the effect of the regularization parameter λ, keeping the variance from exploding, and avoiding overfitting.



(a) OLS



(b) Ridge

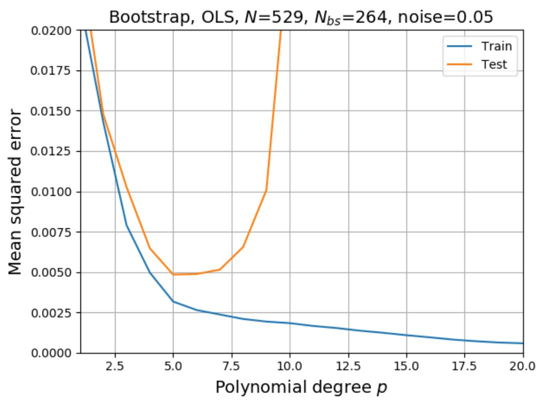


(c) Lasso

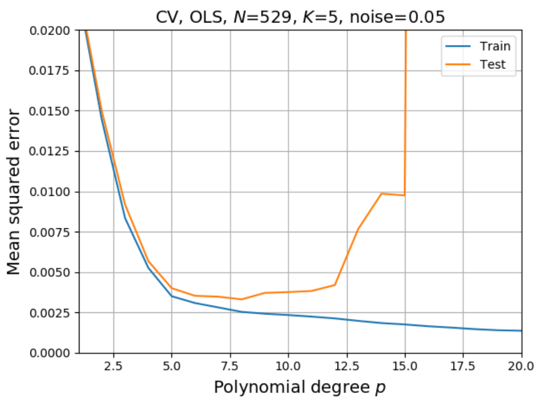
Figure 3: Bias-variance plots for OLS, Ridge, and Lasso regression, where $N = 529$, $\sigma = 0.05$ and $N_{\text{bs}} = N/2$. The best-fit hyperparameter for Ridge was found to be $\lambda = 2.98 \cdot 10^{-5}$, and $\lambda = 3.16 \cdot 10^{-5}$ for Lasso.

Figure 8 in appendix C shows how the OLS results depend on the number of bootstraps. Also in the appendix, figure 9 shows the 95% confidence intervals of β for Ridge regression, which compared to the OLS confidence intervals shown in figure 2 appear to be slightly larger.

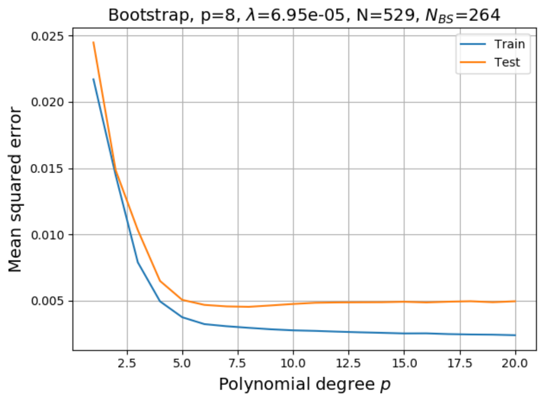
Before looking closer at the hyperparameters λ, we look at the results from the k-fold Cross-validation. Figure 4 shows the difference between the training and test MSE for all regression methods and both Bootstrap and Cross-validation. We see that CV gives very similar results to the Bootstrap, with some discrepancies. For OLS, the test MSE starts shooting up a lot later than for the Bootstrap results. For Ridge, the test MSE lies a lot closer to the training set up until $p = 15$, where it starts going slightly up. Finally, Lasso in general is closer to the training MSE.



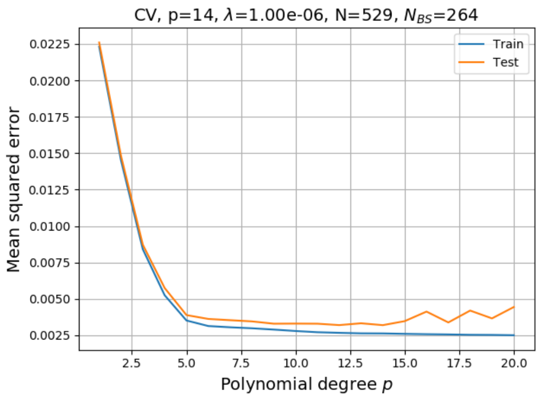
(a) OLS - Bootstrap



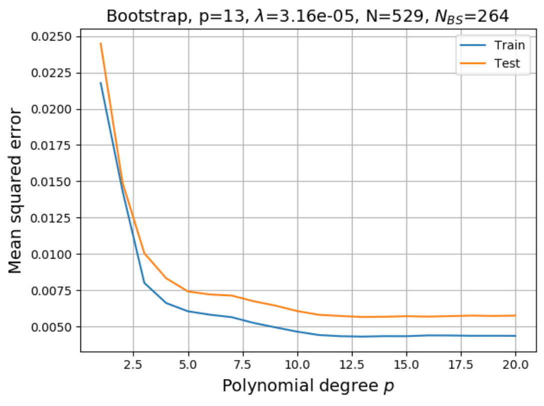
(b) OLS - Cross-validation



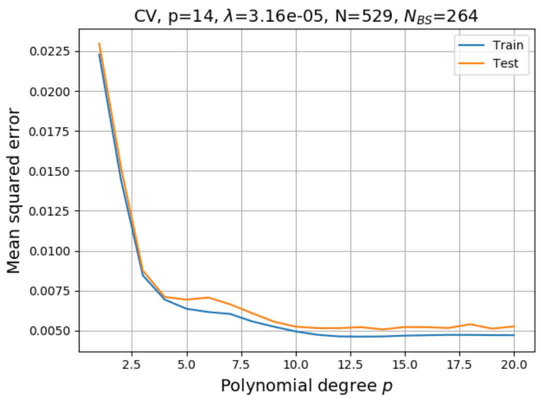
(c) Ridge - Bootstrap



(d) Ridge - Cross-validation



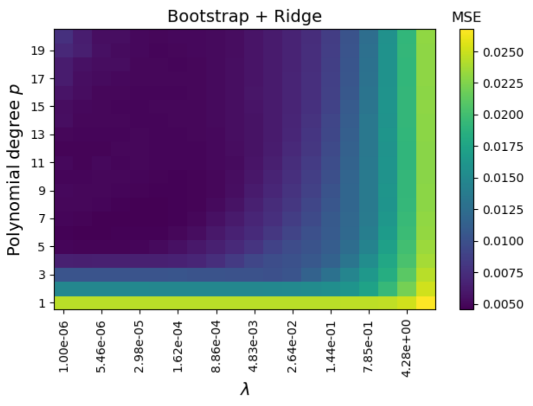
(e) Lasso - Bootstrap



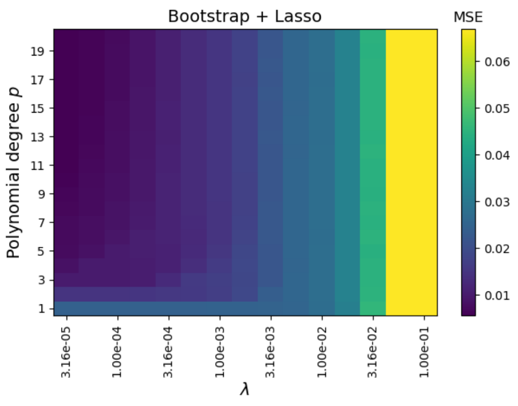
(f) Lasso - Cross-validation

Figure 4: Bias-variance plots for OLS, Ridge, and Lasso regression, where $N = 529$, $\sigma = 0.05$ and $N_{\text{bs}} = N/2$. The best-fit hyperparameter for Ridge was found to be $\lambda = 2.98 \cdot 10^{-5}$, and $\lambda = 3.16 \cdot 10^{-5}$ for Lasso. The p-value on the Ridge + CV panel should have been the p-value corresponding to the minimum MSE value, but for some reason it is wrong.

One way of looking at the hyperparameters would be to look at the heat map from all the values of p and λ that we used. Figure 5 shows the MSE heat map for Ridge and Lasso using Bootstrap. Heat maps for the bias, variance, and MSE from CV can be found in the Github repository [5]. It is clear that the model prefers lower λ, though for high model complexity/polynomial degree a slightly higher λ is preferred. While for Ridge, a large amount of the λs give a good result, for Lasso the MSE almost always increase as λ is increased.



(a) Ridge

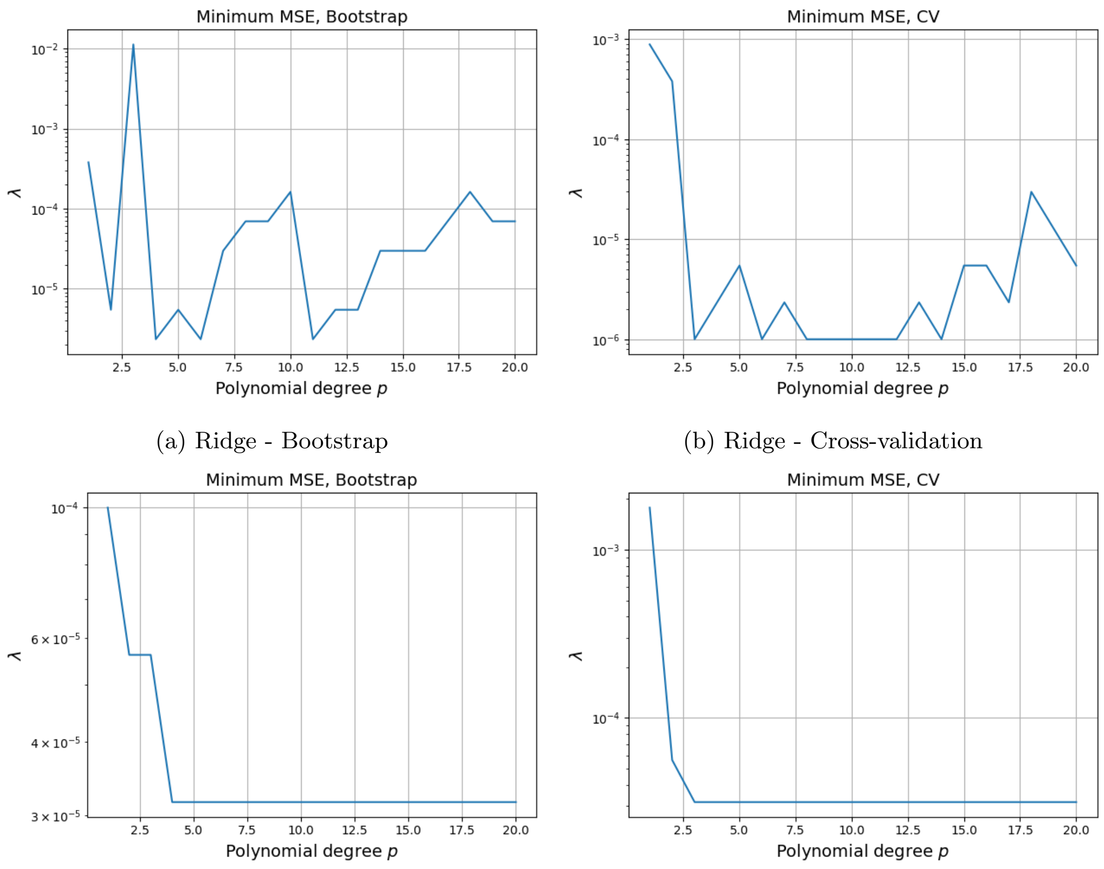


(b) Lasso

Figure 5: Heat maps showing the MSE plotted against polynomial degree p and the hyperparameter λ for both Ridge and Lasso, using Bootstrap resampling.

Figure 6 shows λ_{min} plotted against p for both Ridge and Lasso where λ_{min} is the λ corresponding to the lowest MSE for that polynomial degree. This is done both for Bootstrap and Cross-validation. For Ridge and Bootstrap, the optimal λ varies quite a bit for each value of p, while for CV, it stays at the lowest value for $p \in [8, 12]$, where there is a peak in the Bootstrap. The axes are slightly different, so plotting them on top of each other would have been interesting.

For Lasso, the λ value drops down to the lowest value at $p \sim 3 - 4$ for both Bootstrap and CV, staying there until the end. This shows that we probably should have tried running with even lower values. Sadly, running with the same starting point $\lambda = 1 \cdot 10^{-6}$ as Ridge caused a lot of convergence errors, and took extremely long time without finishing.



(a) Ridge - Bootstrap

(b) Ridge - Cross-validation

(c) Lasso - Bootstrap

(d) Lasso - Cross-validation

Figure 6: Hyperparameter λ plotted against polynomial degree p for Ridge/Lasso regression with Bootstrap/Cross-validation. The λ value corresponds to the one that gives the lowest MSE value at the given p.

Finally, table 2 shows the best-fit parameters p and λ for each of the three regression methods, and for each of the resampling methods. For both Bootstrap and Cross-validation, Ridge regression seems to be the best option, while Lasso seems to be the worst. Part of the reason why Lasso performs the worst here could be because of the λ-values not being low enough as explained earlier, causing us to miss the best fit for Lasso. OLS however performs very close to Ridge, especially in the Cross-validation, and it is the fastest regression method, while Lasso can be extremely slow as there is no analytical solution. That said, both Ridge and Lasso have a significant advantage when it comes to stability. As shown with both the bias-variance trade-off and train-test plots, Ridge and Lasso has a much wider range of parameters that give good results, as the hyperparameter suppresses the overfitting problem with OLS.

Table 2: The best-fit parameters for OLS, Ridge and Lasso regression given the lowest MSE value. First three columns is the Bootstrap, last three is Cross-validation

Reg. method	p (BS)	λ (BS)	MSE (BS)	p (CV)	λ (CV)	MSE (CV)
OLS	5	-	$4.84 \cdot 10^{-3}$	8	-	$3.30 \cdot 10^{-03}$
Ridge	8	$6.95 \cdot 10^{-5}$	$4.55 \cdot 10^{-3}$	15	$5.46 \cdot 10^{-6}$	$3.28 \cdot 10^{-3}$
Lasso	13	$3.16 \cdot 10^{-5}$	$5.67 \cdot 10^{-3}$	14	$3.16 \cdot 10^{-5}$	$5.08 \cdot 10^{-3}$

Before moving on to the terrain data, to point out that for some reason the p and λ values present in the title of figure 4 for the CV+Ridge panel specifically for some reason is incorrect. Quite a bit of time was spent looking at the code and output files without understanding why that one specifically is wrong.

4.2 Terrain data

4.3 Further discussion

maybe refer to how i checked bootstrap compared SKL/OLS, and lecture notes. CV skl vs ols? figure out sectioning Mention bootstrap and how at higher degrees, some samples jst The number of outliers, as in where predicted values values are very far from test data gets higher the higher polynomial degree, and causes the bias to shoot up for some reason Despite resampling, it appears to be the same indices that give the fucked up value In the $n = 1024$, $nbs = 2$, $p = 15$, $noise = 0.05$ case, index 196 has a y_pred of 19 and 25ish setting it to the value of 195 gives a max value of y_pred of 1.2isho why the hell does one row get so incredibly messed up????? And why is it extremely wrong in both bootstraps despite having shuffled everything? Replacing X_test[196] with X_test[195] "removes" the outlier, while just changing y_test[196] does nothing

Maybe related to RNG and seed? WATHa haw results are weird with minimum in files

5 Conclusion

Introduce why we set out, then explain results

no idea why problems are, but the way the code is structured, sadly makes debugging slow.

References

[1] Richard Franke. A critical comparison of some methods for interpolation of scattered data, 1979.

[2] Earthexplorer. `https://earthexplorer.usgs.gov/`.

[3] Hastie et al. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction.* Springer, second edition edition, 2017.

[4] Wessel N. van Wieringen. Lecture notes on ridge regression, 2020.

[5] Github repository, project 1. `https://github.com/simennb/fysstk4155-project1`.

Appendix

A Data sets

A.1 Franke function

The Franke function as given in [1]

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

where $x, y \in [0,1]$.

A.2 Terrain data

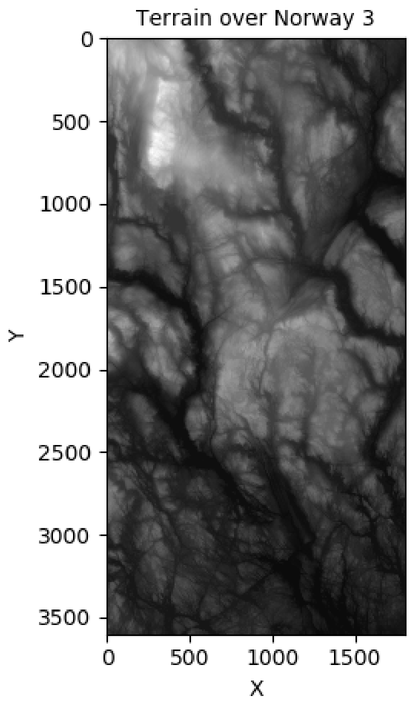
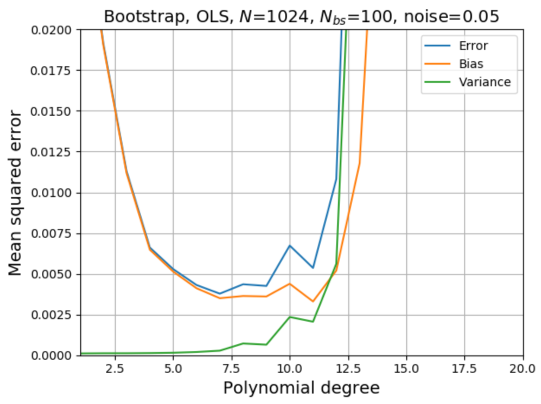


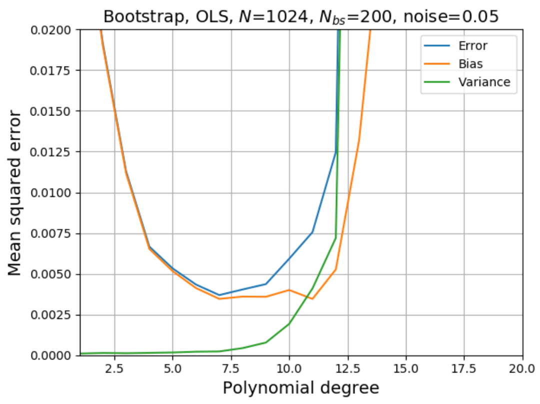
Figure 7: Plot of the entire terrain map in `SRTM_data_Norway_3.tif`. Map is over a region south-east in Norway (Vestfold og Telemark, Viken). Coordinates $(x,y) = (1100, 3000)$ correspond to the location of the city Skien.

B Testing and verifying methods

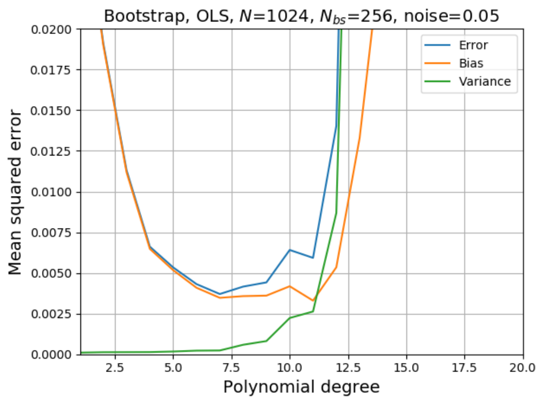
C Franke function plots



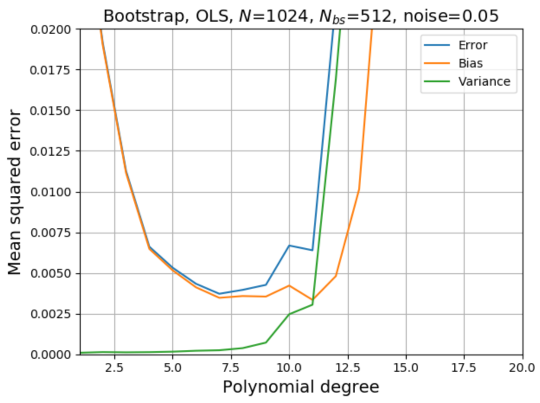
(a) $N_{\mathrm{bs}} = 100$



(b) $N_{\mathrm{bs}} = 200$



(c) $N_{\mathrm{bs}} = 256$



(d) $N_{\mathrm{bs}} = 512$

Figure 8: Bias-variance plots for OLS where the number of bootstraps N_{bs} is varied. $N = 1024$, $\sigma = 0.05$. In general, all four produce very similar results.

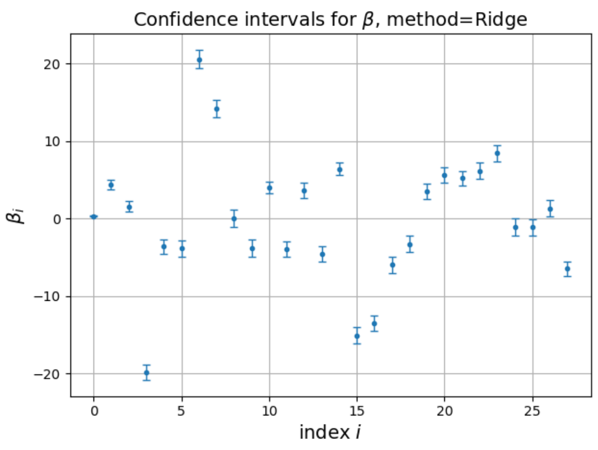


Figure 9: 95% confidence intervals for the regression coefficients β_{Ridge} for $p = 6$, $N = 529$, and noise $\sigma = 0.05$.

D Terrain data plots