Regression Analysis and Resampling Methods

Oda Hovet, Ilse Kuperus, & Erik Alexander Sandvik

October 10, 2020

Abstract

We implement three linear regression methods for assessment; Ordinary Least Squares (OLS), Ridge regression and Lasso regression. We implement the methods on both self-generated data using Franke's function, and on real terrain data of the west coast of Norway. We make assessments of the linear regression methods using the mean squared error computed using the resampling techniques K-fold Cross Validation and Bootstrap. We demonstrate that Ridge regression perform the best on the self-generated data, although we are not able to rule out that the resource expensive Lasso regression method have potential to perform better. On the terrain data we see that Lasso yields the best result, however even for the best results this gives a minimum mean squared error of the test data of about 8900.

Contents

1 Introduction

Linear regression describes the relationship between some observations and the predictors of the observations. The relationship between dependent and independent variables can be given by a function which can be modelled. In this project, we explore the three regression methods Ordinary Least Squares (OLS), Ridge and Lasso for predicting a model to data. We are going to test these methods on the Franke function and real-terrain data to determine which method is best suited. The models will be evaluated based on the mean squared error (MSE) and the R^2 score. A central analysis is the bias-variance decomposition of the mean squared error. For selecting the best model, we apply the resampling techniques bootstrap and k-fold cross-validation. After introducing the different methods and seeing how we can optimize them we will look at some real terrain data and use the methods on this data to illustrate how the methods can be applied to real datasets.

This project is structured into three main sections. The relevant theory and methods we will be using are presented in section 2. In section 3 we provide the results obtained from the different methods and discuss these results and their consequences. Lastly, a conclusion is presented in section 4.

2 Theory & Methods

2.1 Linear Regression Methods

Given a set of p inputs $\mathbf{x}^T = (x_0, x_1, \ldots, x_{p-1})$ and a corresponding output y, we make a prediction $\tilde{y}$ of y by a linear model

$$\tilde{y} = \sum_{j=0}^{p-1} x_j \beta_j$$

where $\beta_j \in \mathbb{R}$ are unknown coefficients. Then we can write the output on the form

$$y = \tilde{y} + \epsilon$$

where ϵ is the error. If we collect the coefficients into a vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{p-1})$ we can write the model as an inner product

$$\tilde{y} = \mathbf{x}^T \boldsymbol{\beta}$$

When doing polynomial regression we simply replace the x_i's by x, x^2, and so on. To determine the unknown coefficients we need a set of $n \geq p+1$ inputs $\left\{\mathbf{x}_i^T\right\}$ and a set of corresponding outputs $\{y_i\}$ for $i = 0, \ldots, n-1$. If we collect the inputs in an $n \times p$ matrix $\mathbf{X}$ (often called *the design matrix*) where $\mathbf{x}_i^T$ is the i-th row of $\mathbf{X}$ we can predict the vector of outputs $\mathbf{y}$ by

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$$

The goal now is to choose the coefficients vector $\boldsymbol{\beta}$ such that $\tilde{\mathbf{y}}$ becomes the best possible prediction of $\mathbf{y}$. Obviously, we'd like to have $\tilde{\mathbf{y}} = \mathbf{y}$, but this depends on whether $\mathbf{y}$ lies in the column space of $\mathbf{X}$. In general this is not possible, so what then is considered "the best possible prediction" of $\mathbf{y}$?

The answer varies between linear regression methods. According to the method of OLS, the best possible prediction is generated by the coefficient vector that minimizes the "cost function"

$$C^{\text{OLS}}(\boldsymbol{\beta}\,|\mathbf{X}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2$$

According to the method of Ridge regression, it is generated by the coefficient vector that minimzes the cost function

$$C^{\text{Ridge}}(\boldsymbol{\beta}\,|\mathbf{X},\lambda) = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2 + \lambda\sum_{j=0}^{p-1}\beta_j^2$$

where $\lambda \geq 0$ is a so-called "penalty parameter". Finally, according to the method of LASSO regression, the best possible prediction is generated by the coefficient vector that minimizes the cost function

$$C^{\text{LASSO}}(\boldsymbol{\beta}\,|\mathbf{X},\lambda) = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2 + \lambda\sum_{j=0}^{p-1}|\beta_j|$$

The minimization problem of OLS and Ridge have the analytical solutions (see the Appendix for derivations)

$$\hat{\boldsymbol{\beta}}^{\text{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

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

where $\mathbf{I}$ is the $p \times p$ identity matrix. In general, $\mathbf{X}$, and in consequence $\mathbf{X}^T\mathbf{X}$, may not be invertible. This happens when there are multiple solutions to the minimization problem of $C^{\text{OLS}}(\boldsymbol{\beta}\,|\mathbf{X})$. In that case we can replace $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ by the so-called *pseudo-inverse* $\mathbf{X}^+$ of $\mathbf{X}$. The pseudo-inverse can be constructed from a singular value decomposition of $\mathbf{X}$, and it can be shown that $\hat{\boldsymbol{\beta}}^{\text{OLS}} = \mathbf{X}^+\mathbf{y}$ is the global minimum of $C^{\text{OLS}}(\boldsymbol{\beta}\,|\mathbf{X})$ [1]. So this expression is what we actually use in practice.

Since $|\beta_j|$ is not differentiable the minimization problem of $C^{\text{LASSO}}(\boldsymbol{\beta}\,|\mathbf{X}, \lambda)$ has no analytical solution. Instead the minimum must be found numerically.

Since we're adding a little bit of randomly distributed noise to the Frankefunction, drawn from a Gaussian distribution with zero mean and variance σ^2, the elements of $\mathbf{z}$, $z_i = f(x_i, y_i) + \epsilon_i$, are also randomly distributed. If we then use $\mathbf{z}$ to calculate $\hat{\beta}$ the elements of $\hat{\beta}$ are randomly distributed as well. We show in the appendix that, when using OLS, the variance of $\hat{\beta}_j^{\text{OLS}}$ is given by

$$\text{Var}\left(\hat{\beta}_j^{\text{OLS}}\right) = \sigma^2 \left(\mathbf{X}^T\mathbf{X}\right)^{-1}_{jj}$$

This can be used to find the confidence interval of $\hat{\beta}_j^{\text{OLS}}$. With OLS the mean of $\hat{\beta}_j^{\text{OLS}}$ is the same as what we would get if there were no noise. This can be shown by using the expression for $\hat{\beta}^{\text{OLS}}$and using that the noise has zero mean. Although we are the ones generating the data set, we pretend that we have no control over the noise, so the mean of $\hat{\beta}^{\text{OLS}}$is unknown in that sense. Since the data set is fixed, we cannot approximate the true mean by a sample mean. So when finding the confidence intervals we have no choice but to approximate the true mean by whatever we get for $\hat{\beta}_j^{\text{OLS}}$.

Since we pretend that we have no control over the noise, σ^2 is also unknown, so it must be estimated. We use the following estimate

$$\sigma^2 \approx \frac{1}{n-p-1}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2$$

which can be found in Hastie et al [2].

2.2 Model Assessment and the Bias-Variance Decomposition

Since a polynomial of degree p is uniquely determined by $p+1$ points on the curve, one can for any data set (x_i, y_i) find a polynomial which fits this data set perfectly. If $\mathbf{y}$ is the output data vector and $\tilde{\mathbf{y}}$ is the prediction of the output data generated by some linear regression method, then we can use e.g. the mean squared error

$$\text{MSE}\left(\mathbf{y},\tilde{\mathbf{y}}\right)=\frac{1}{n}\sum_{i=0}^{n-1}\left(y_i-\tilde{y}_i\right)^2$$

or the R^2 score

$$R^2\left(\mathbf{y},\tilde{\mathbf{y}}\right)=1-\frac{\sum_{i=0}^{n-1}\left(y_i-\tilde{y}_i\right)^2}{\sum_{i=0}^{n-1}(y_i-\overline{y})^2}$$

where $\overline{y}=\sum_{i=0}^{n-1}y_i/n$, as measures of how well the linear regression method is performing. When increasing the complexity of the linear model (i.e the polynomial degree) the MSE will tend towards zero, while the R^2 score will tend towards 1, regardless of what linear regression method we use. So any regression method can apparently be made arbitrarily good by increasing the complexity. However, when trying to use this polynomial as a model or a prediction of new data, it's probably going to fit the new data very badly. This is called *overfitting*. Instead, to properly assess the respective regression methods, we split the inital data set into two parts; the *training data* and the *test data.* The training data is what we use to create a model using a linear regression method, while the test data is used to assess the method by calculating the MSE and R^2 score. About $20-25\%$ of the data set is chosen as test data. The goal is then to find the model complexity such that e.g. the MSE for the test data is at a minimum.

However, ideally, we'd like the model to perform well on *any* test data, not just the selected test data. Actually, we'd like the *method* to generate, from any data set, a model which performs well on any other data set. Since we don't have every possible data set to check that this is the case, we can use so-called resampling techniques like the Bootstrap technique or Cross Validation as a cheap way to create several data sets out of the one data set that we have.

Another two important quantities when making an assessment of the regression method are the *bias* and the *variance*. In the appendix we show that the MSE can be written on the form

$$\text{MSE}\left(\mathbf{y},\tilde{\mathbf{y}}\right)=\frac{1}{n}\sum_{i=0}^{n-1}\left(f_i-\mathbb{E}\left[\tilde{y}_i\right]\right)^2+\frac{1}{n}\sum_{i=0}^{n-1}\left(\tilde{y}_i-\mathbb{E}\left[\tilde{y}_i\right]\right)^2+\sigma^2$$

where the first term on the right side is the bias, the second term is the variance and the third term is the variance of the noise. The bias is the difference between the true function and the

expectation value of the model, thus this is the error in our model itself. This error will decrease as we have a higher complexity because our model will be better fit to the data. The variance is the difference between the model and the expectation value in the model, thus this is the fluctuation in the model. As we increase the complexity in our model the variance will increase as the model will now better fit the data, but will also fit the noise leading to larger fluctuations in the variance. The variance of the noise will also be a part of the MSE as the fluctuations in the noise will change the model and therefore change the MSE.

2.3 Resampling techniques

The first resampling technique we'll be using is the Bootstrap method. Given an array of data of length n, assumed to be drawn from some distribution, the Bootstrap method can be used to estimate any property of this distribution, like the mean or standard deviation. We pick a positive integer B, the "number of bootstraps", and for each B we draw n elements from the data set at random. The same element may be drawn more than once up to n times. This way we end up with B arrays of n elements. To estimate a property of the underlying distribution, we can calculate the same property of each array, and then take the average over all B arrays.

Another resampling technique is the so-called K-fold Cross Validation (CV) method. In contrast to the Bootstrap method where the same data set is resampled for each bootstrap, the CV method splits the data in K parts. Also, while the Bootstrap method is mainly used to estimate any statistic of some data set, the CV method is mainly used to assess the performance of regression methods, although you can easily implement the CV method to estimate statistics of data sets and Bootstrap to assess regression methods.

The CV method goes as follows: We select one part as the test data and the other $K-1$ parts as training data. We then use the training data to make a prediction of the testing data using some regression method, and compare the outcome with the actual testing data, perhaps by calculating the mean squared error or the R2-score.

We then repeat this process another $K-1$ times such that each part of the data set is used as testing data once and as training data $K-1$ times. What we end up with is K values of, say, the mean squared error. Finally we simply take the average of these values, which is the final "score" of the regression method that reflects its performance.

2.4 The Franke's Function

Franke's bivariate test function, commonly just called Franke's function, is a two-dimensional widely used test function in interpolation and fitting problems. It is the weighted sum of four exponentials

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

where $x, y \in [0, 1]$. As can be observed in Figure 1, the surface of the plotted Franke's function has two Gaussian peaks and one Gaussian dip. The x and y data points are scattered in pairs in space. If the data points are random, it should be determined which are "nearby" by sorting the data [3].

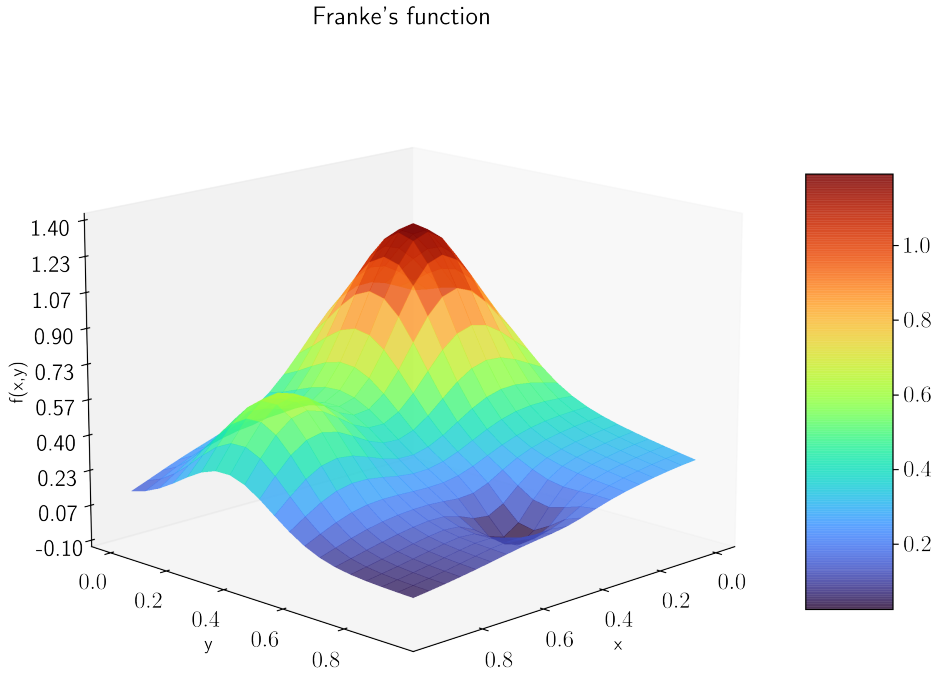


Figure 1: Franke's bivariate test function given in Equation 1 is defined for $x, y \in [0, 1]$.

3 Results and Discussion

3.1 Data From The Franke Function

3.1.1 MSE and R^2 Scores Using OLS

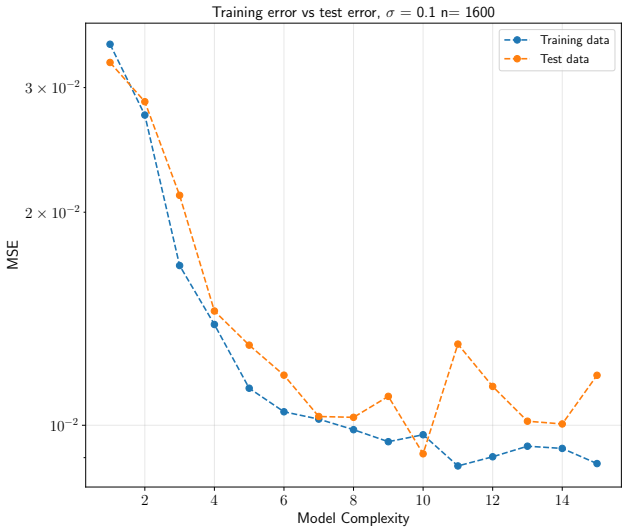
The data generated from the Franke function was split into training (80%) and test (20%) data. Table 1 displays the training data of the MSE and R^2 scores from applying OLS with polynomials x and y up to the fifth order with normal and random noise. The table displays the varying results from 100 and 10000 data points and the noise σ^2 set to 0.01 and 1.

Table 1: Training MSE and training R^2 scores by applying OLS using 100 and 1000 datapoints with σ^2 set to 0.01 and 1.

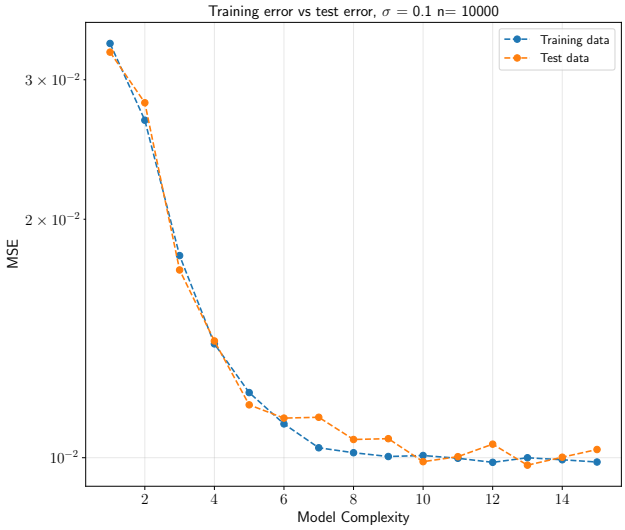
n	σ^2	MSE	R^2
100	0.01	0.009	0.889
100	1	0.550	0.281
10000	0.01	0.012	0.870
10000	1	1.004	0.086

It can be observed in Table 1 that a small σ^2 of 0.01 results in small MSE and high R^2 scores for both low and high number of data points. This indicates a good fit of the response data. With more noise of $\sigma^2 = 1$, the MSE and R^2 scores increase and decrease drastically, respectively, and do not explain the data well.

Figure 2 shows the mean-squared error of the test and training data as a function of model complexity. In our case, the model complexity is the degree of the polynomial used in the design matrix.



(a)



(b)

Figure 2: MSE for both training and test data as a function of model complexity. The plot was made with OLS with 1600 data points in (a) and 10 000 data points in (b). Both with noise with a standard deviation $\sigma = 0.1$

Figure 2(a) shows that the MSE becomes smaller as the complexity increases for the training data, but does not generalize as well for the test data. This is an example of overfitting where the model becomes too complex and flexible and when the test data is included, there is an overfit in the training data that does not generalize to the test data. The variance is higher in the model.

When increasing the complexity further the overfitting will continue and the mean squared error in the test data will start to increase. The data was scaled to remove outliers. It is favorable with a model that is close to the training data with low MSE between the model and the training data. The training and test data get more similar with 10000 data points (Figure 2(b)) compared to 1600 data points Figure 2(a).

3.1.2 Bootstrap and Cross-Validation of OLS

We have used the bootstrap resampling method with 400 resampling points. This we have used to calculate the bias, variance and total mean squared error as a function of model complexity.

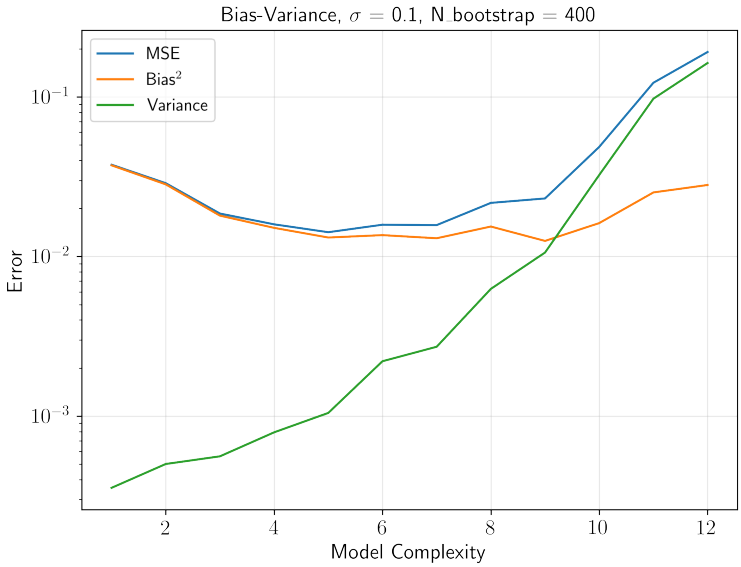
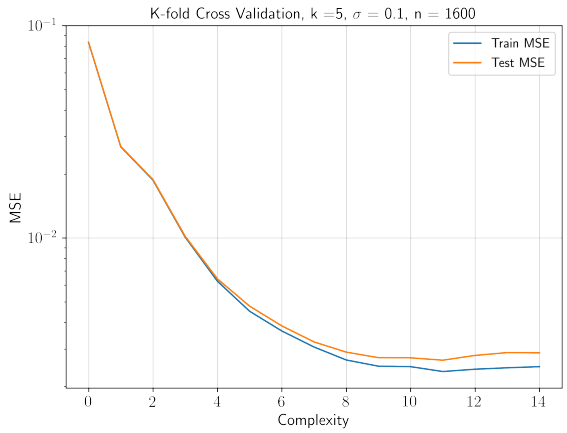


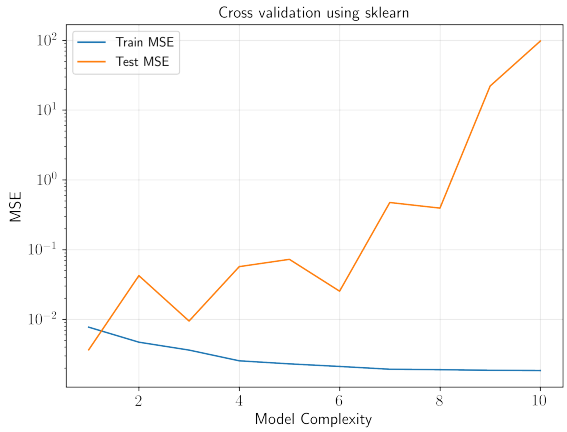
Figure 3: Bias-variance trade-off (error) as a function of model complexity from using 400 bootstraps on OLS. With added noise with $\sigma = 0.1$ and 1600 data points.

In Figure 3 we see the plot of the bias variance trade-off. We see from this plot that the squared bias starts high and decreases with complexity while the variance starts low and increases with complexity like expected. In addition, we see that the MSE is the sum of the bias and the variance. The optimal complexity is the complexity where we have the lowest MSE, which in this case would be a polynomial of about degree 5. In this image we also note that at a high complexity the squared bias goes up a little again instead of continuing on a downwards trend. This might be because the error in the model generally increases when we reach a high enough complexity.

The other resampling technique we used is k-fold cross validation. Figure 4 shows the MSE train and test data as a function of model complexity for the OLS method using a 5-fold cross-validation.



(a) K-fold cross validation with n = 100 datapoints and noise with $\sigma = 0.1$



(b) K-fold cross validation with n=1600, and noise with $\sigma = 0.1$

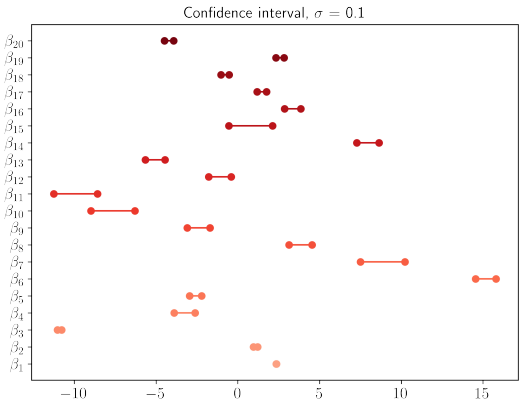
Figure 4: Cross Validation on OLS

From Figure 4 we can see the MSE for the test and train data as a function of model complexity for a different number of data points. We see in Figure 4a that when we have 100 data points the test MSE will increase when complexity gets higher and the train MSE will decrease. This is due to overfitting, where the model learns the data well due to the complexity, but the model does not generalize well and will thus get a higher error from the test data. In Figure 4b we see the cross validation with more data points. This illustrates that when we use more data points we will be able to have a higher complexity before we get overfitting.

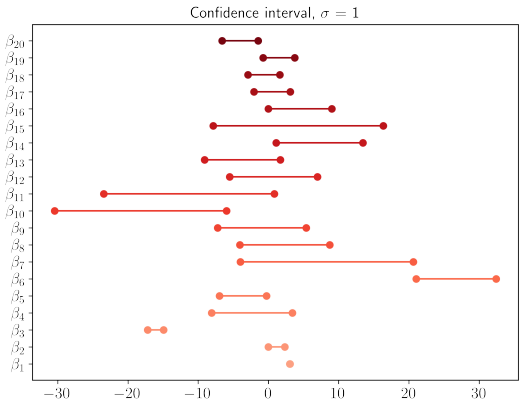
The cross validation method will decrease the overfitting slightly from when we used no resampling techniques. This is because of the fact that we use a different data set to make the model and predict using different test values and average over the MSE of all these. Because of the change in the data set the model will not be able to get as close to the data set and overfitting will be reduced.

3.1.3 Confidence Intervals of OLS

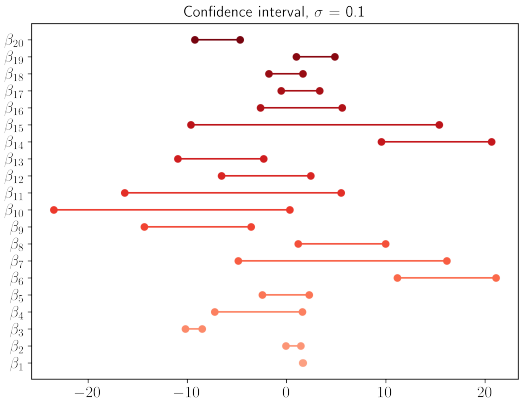
The 95% confidence intervals were calculated for complexity $p = 5$ for $n = 100$ and $n = 10000$ data points and $\sigma^2 = 0.01$ and $\sigma^2 = 1$ presented in Figure 5(a) – (d).



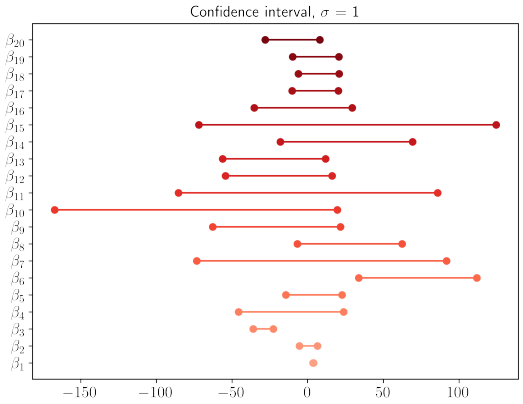
(a) Confidence intervals for n = 10000 $\sigma^2 = 0.01$



(b) Confidence intervals for n = 10000 $\sigma^2 = 1$



(c) Confidence intervals for n = 100 $\sigma^2 = 0.01$



(d) Confidence intervals for n = 100 $\sigma^2 = 1$

Figure 5: Confidence intervals of 95 % for β_1 to β_{20} with complexity of $p = 5$ using OLS. Using different amount of datapoints n and different variance σ^2

Overall, the β intervals are substantially narrower for 10000 data points in (a) and (b) compared to 100 in (c) and (d). This is understandable since the more data points included, the more confident is the estimate of each parameter. The shrinkage of the intervals displayed in Figure 5(a) and (b) is also due to the increase of data. The different σ^2 as 0.01 and 1 in (c) and (d), respectively, hardly generates any differences in the two plots, and is more different in (a) and (b) for $n = 10000$.

Generally, the lower degree parameters are narrower and therefore more significant than the higher degree parameters. This can be explained given that the shortest intervals contain the simplest parameters of x and y. The higher degree parameters β_7 to β_{15} are broader where many include 0 inside the CI which can raise questions regarding their relevance for the model. The highest parameters from β_{16} to β_{20} are narrower again and more significant. A higher σ^2 in (b) also gives broader intervals with index from -30 to 30. The intervals in (b) are therefore more widely distributed than in (a) which has lower σ^2.

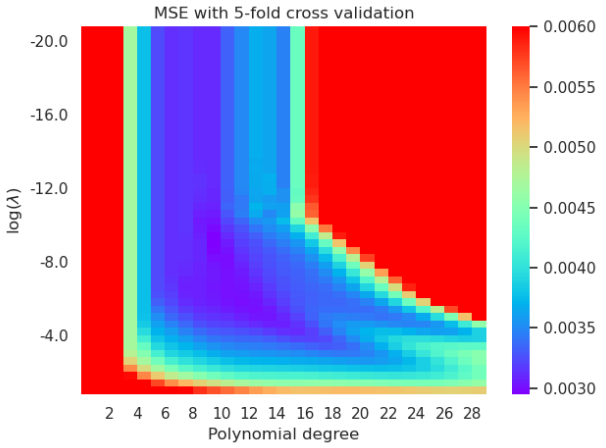
3.1.4 Bootstrap and Cross-Validation of Ridge

Figure 6 shows the MSE, bias and variance as a function of the penalty parameter λ and the complexity of the model. The MSE is calculated using both CV and the Bootstrap techniques, shown in Figure 6a and Figure 6b, while the bias and variance is calculated using Bootstrap, shown in Figure 6c and Figure 6d. In Figure 6a and Figure 6b we see that the respective resampling techniques to a large extent agree on the MSE. We obtain a somewhat more localized minimum in the MSE with CV at MSE $= 2.9 \cdot 10^{-3}$, but the minimum in the MSE calculated using Bootstrap is somewhat lower at MSE $= 2.2 \cdot 10^{-3}$ There is however a major disagreement about what the MSE should be in the upper right section of the plot. In this section Bootstrap says that the model performs only $\sim$ 1.3 times worse than at minimum, while CV says that the model performs > 2 times worse than at minimum. Actually, the largest MSE calculated using CV is ~ 10 times higher than at minimum, which clearly suggest that there has been an overfitting.

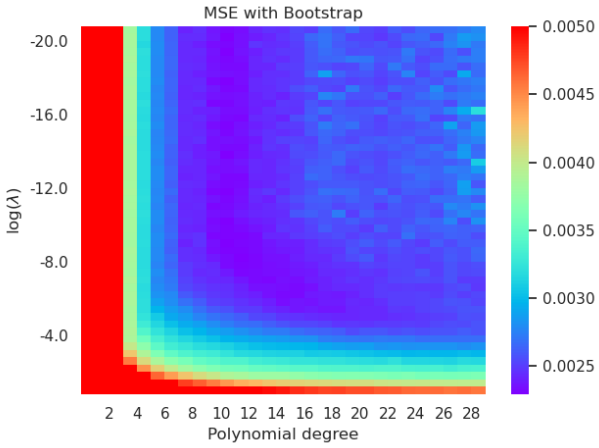
There is also a contradiction in Figure 6b, Figure 6c, and Figure 6d with the bias-variance decomposition. According to the bias-variance decomposition, the MSE is the sum of the bias and the variance (as well as the variance of the noise). In these figures this is clearly not the case as the bias exceeds the MSE everywhere, while the variance is comparable to the MSE. This might suggest that there is a major fault in using Bootstrap resampling as an assessment method of linear regression techniques. While CV splits the data set into K folds and keeps them separate, the Bootstrap method resamples the same data set over and over again. Depending on the initial data set, we may get B data sets that are very similar which leads to a low variance and a (potentially) high bias. Unfortunately, we do not have bias and variance calculated using CV which could've been compared to the bias and variance calculated using Bootstrap.

Both Bootstrap and CV agree however that Ridge regression performs better than OLS on the data set generated from the Frankefunction. We see in Figure 3 and Figure 4 that the

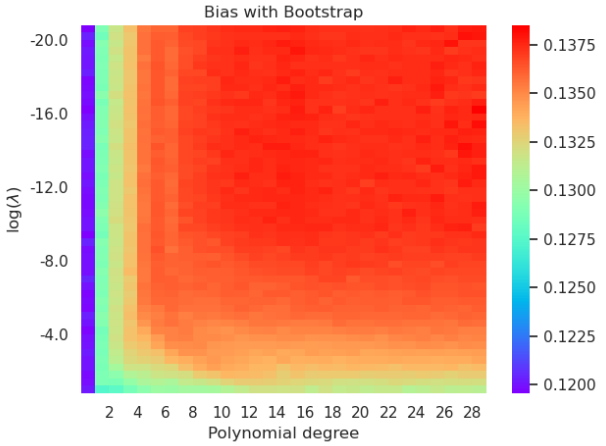
minimum of the MSE from OLS is of the order $\sim 10^{-2}$, while the minimum of the MSE from Ridge is of the order $\sim 10^{-3}$.



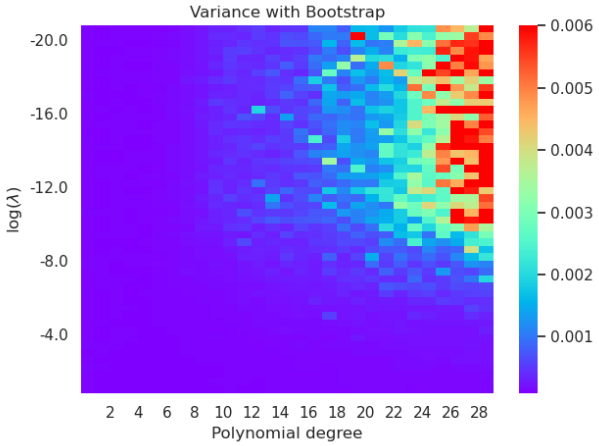
(a) MSE with the CV resampling technique



(b) Ridge MSE with the Bootstrap resampling technique



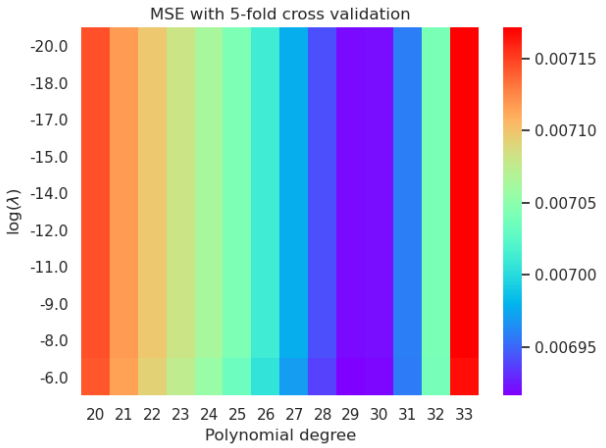
(c) Bias with the Bootstrap resampling technique



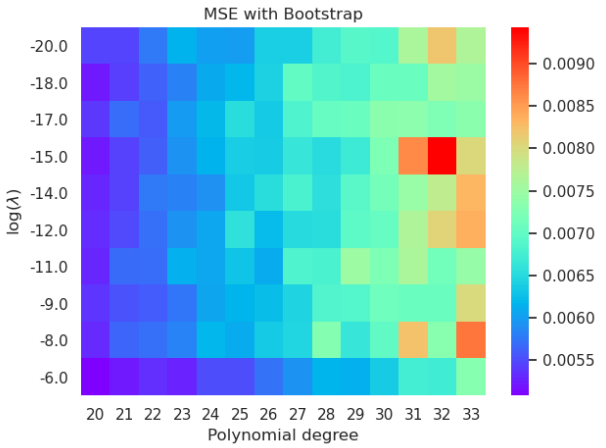
(d) Variance with the Bootstrap resampling technique

Figure 6: MSE of the prediction of the test data calculated using both the CV and the Bootstrap resampling techniques as well as bias and variance calculated using Bootstrap.

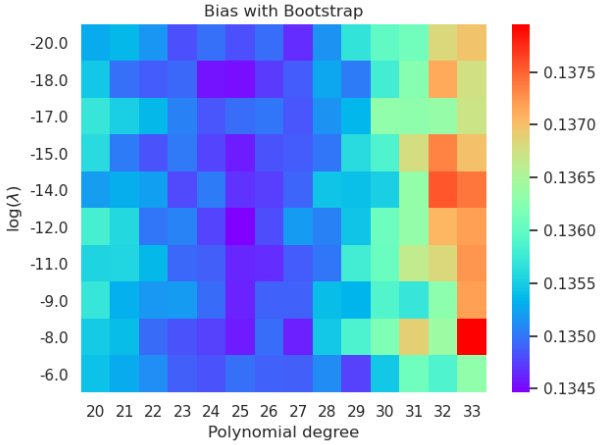
3.1.5 Bootstrap and Cross-Validation of Lasso



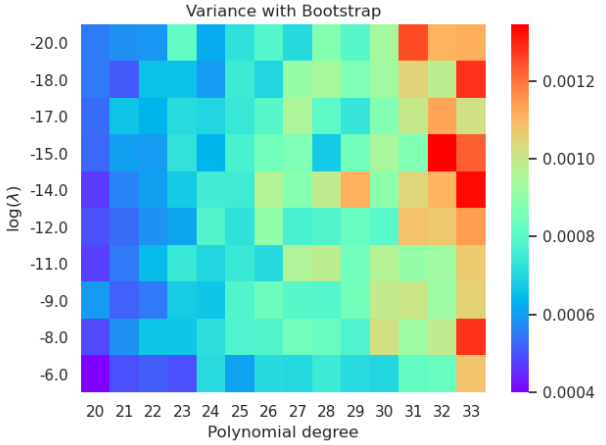
(a) MSE with the CV resampling technique



(b) MSE with the Bootstrap resampling technique



(c) Bias with the Bootstrap resampling technique



(d) Variance with the Bootstrap resampling technique

Figure 7: Lasso MSE of the prediction of the test data calculated using both the CV and the Bootstrap resampling techniques as well as bias and variance calculated using Bootstrap.

Figure 7 shows the MSE calculated using CV and Bootstrap as a function of the penalty parameter λ and the model complexity. The bias and variance calculated with Bootstrap is also shown. We see a rather striking disagreement on the MSE between CV and Bootstrap. According to CV, the best model of the data is a polynomial of degree ~ 30, while according to Bootstrap, the polynomial degree should be < 21. Having already argued why Bootstrap

might be a faulty resampling technique compared to CV, it is actually this resampling technique that yields results that agrees the most with the MSE calculated using Ridge regression, where the MSE was lowest for a polynomial of degree ~ 10. It's also interesting that according to CV, the MSE is not dependent on the penalty parameter λ for a given polynomial degree. There is also no clear pattern in Figure 7b that suggests that the penalty parameter should be smaller or larger in order to minimize the MSE.

The MSE for lower polynomial degrees than $p = 20$ were not investigated. The implementation of Lasso regression in Scikit-Learn uses gradient descent to find the minimum of the cost function, which is computationally taxing. Thus a smaller interval of polynomial degrees and values of λ had to be selected. In order to find an interval where the MSE was at a minimum the CV resampling technique was used. We thus leave for future study the investigation of the MSE calculated using Bootstrap to see if this agrees with the results we obtained for Ridge regression.

All in all we obtain a higher MSE using Lasso regression than with Ridge regression. The main reason is the choice of tolerance for the optimization, at $\text{tol} = 10^{-3}$. Our experience is that a lower tolerance minimizes the MSE even further, although we are not able to give a proper demonstration of this due to our current lack of access to proper computational equipment, since a lower tolerance will make Lasso regression even more computationally taxing. We are thus not able to rule out that Lasso can perform better than Ridge regression.

As for the bias and the variance, also here did we not get results in agreement with the Bias-Variance Decomposition, as the calculated bias is significantly higher than the MSE.

3.2 Data From The Digital Terrain

With the implementation of OLS, Ridge and Lasso regression on data generated from the Franke function, we are now going to apply the three methods of linear regression on real data of a digital terrain. The digital terrain is from a region close to Stavanger, Norway and can be observed in Figure 8. We will use cross-validation as the resampling method to evaluate which model that fits the data best.

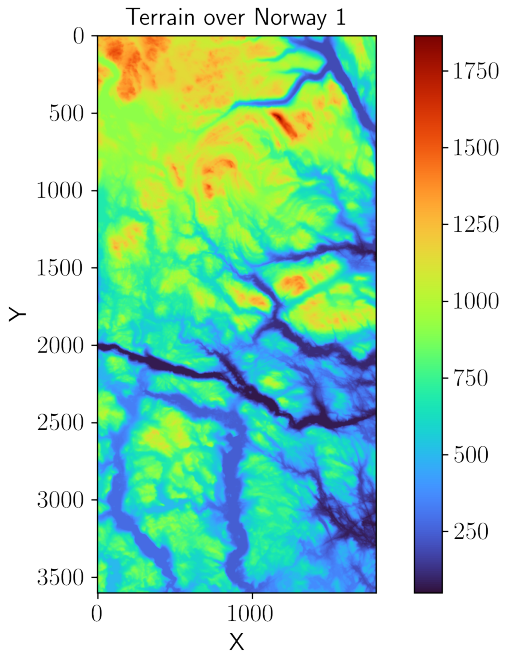


Figure 8: Terrain data visualized with colors. The color bar represents the height in the terrain.

3.2.1 OLS on Terrain Data

A 5-fold cross-validation with OLS was performed on the terrain data and is presented in Figure 9. For this cross validation we have used slice index 19 which gives us 18050 datapoints.

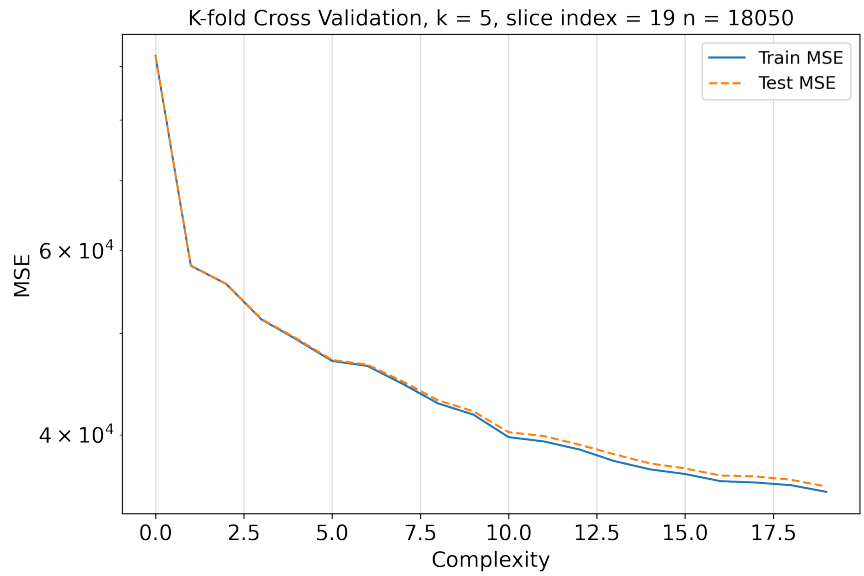


Figure 9: 5-fold cross validation with OLS on terrain data with MSE as a function of increasing complexity.

We have performed the cross validation for polynomial degree up to 20. From Figure 9 we see that for both the test and train MSE get lower for higher complexity. We see that this curve is similar to the curve we got using 5-fold cross validation on the Franke function when using a similar amout of datapoints. We see that this model can fit the terrain data up to at least a 30th order polynomnial without any major overfitting and we get a MSE of about 35710. Since we have quite a lot of data in this case we do not have a lot of overfitting. With less data, outliers could have had more influence on the overall fit and would have resulted in more overfitting of the model.

3.2.2 Ridge Regression on Terrain Data

We also apply the ridge regression on the terrain data. Here again we use k-fold cross validation for resampling. A 5-fold cross-validation with Ridge was performed on the terrain data with MSE as a function of $\log_{10}(\lambda)$ for lambda values -12 to 2. We chose the four different complexities; 3, 5, 8 and 11 and made a plot for each of them as shown in Figure 10.

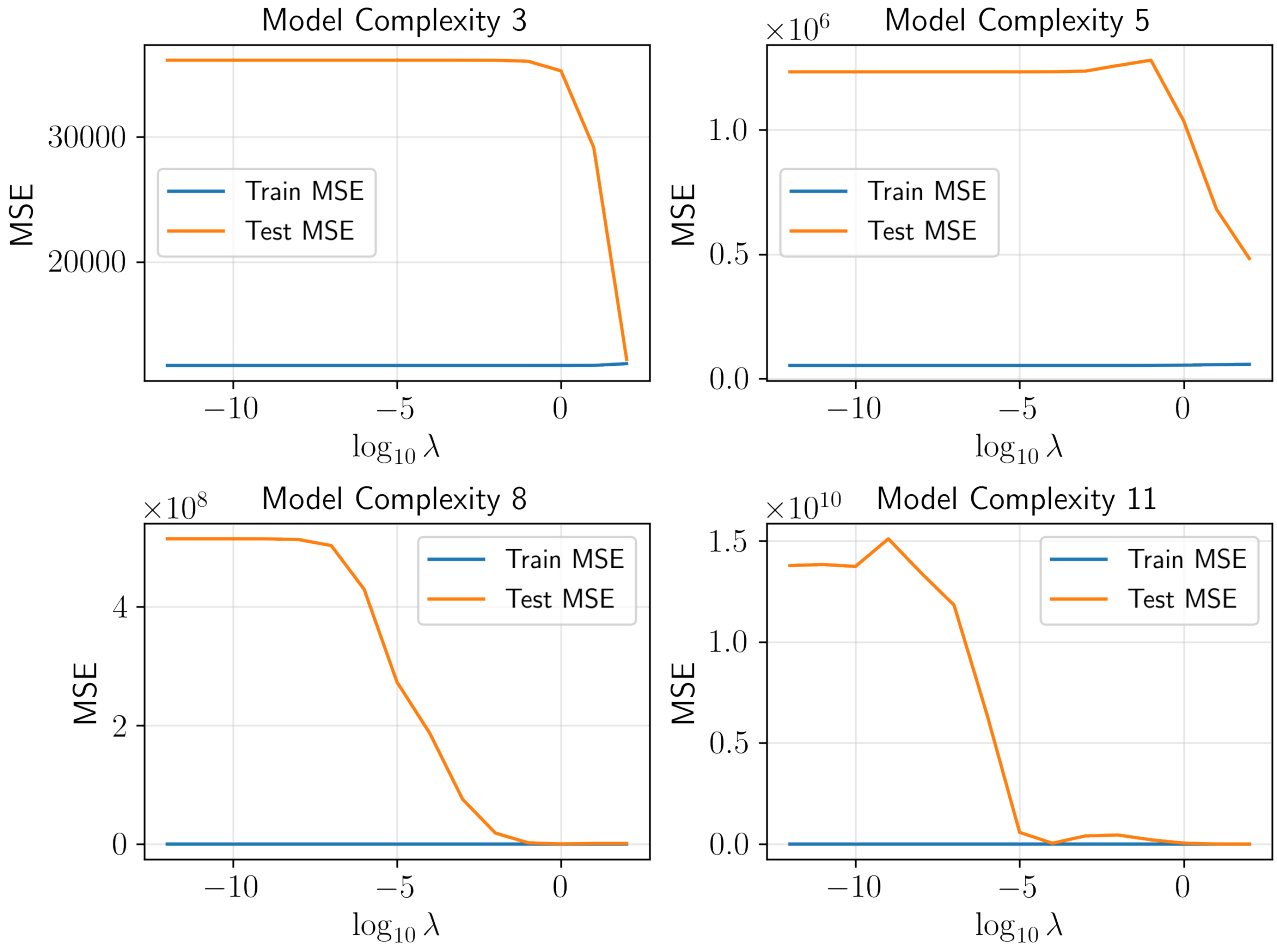


Figure 10: 5-fold cross validation with Ridge on terrain data with MSE as a function of $\log_{10}(\lambda)$. The cross validation was performed for complexities 3, 5, 8 and 11.

The data points were sliced with index 10. For the four plots in Figure 10, we can observe overfitting. The higher the value of λ is, the more regularization and less flexibility it has as it penalizes more and thus goes towards 0. The higher model complexity, the higher the initial MSE is before penalization causes it to go towards 0.

3.2.3 Lasso Regression on Terrain Data

For Lasso regression, we also performed a 5-fold cross-validation on the terrain data with MSE as a function of $\log_{10}(\lambda)$ for lambda values -12 to 2. We chose the four different complexities; 3, 5, 8 and 11 and made a plot for each of them as shown in Figure 11.

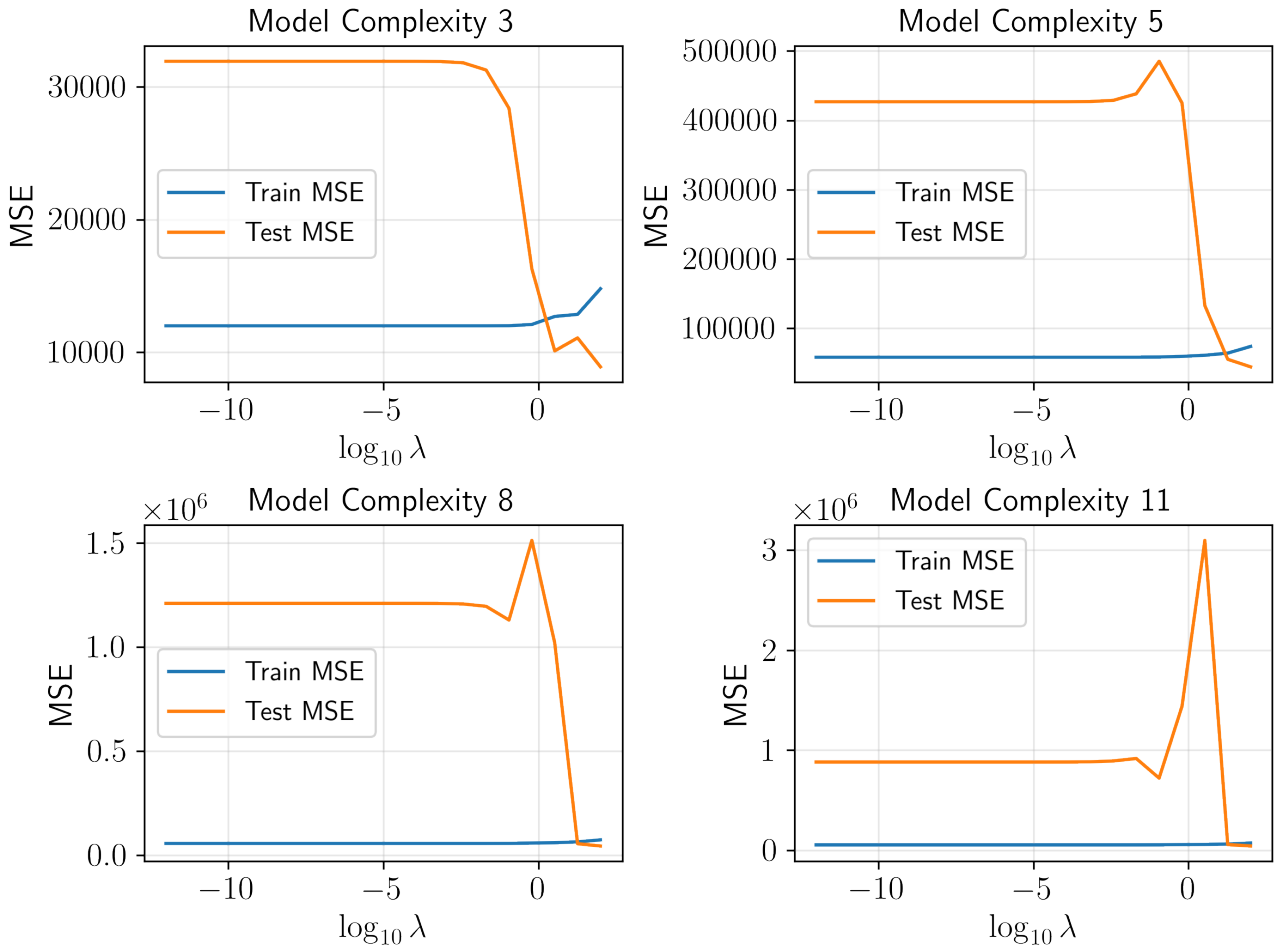


Figure 11: 5-fold cross validation with Lasso on terrain data with MSE as a function of $\log_{10}(\lambda)$. The cross validation was performed for complexities 3, 5, 8 and 11

The data points were sliced with index 50 for computational concerns. In Figure 11 we see that for the terrain data a λ of about 1 gives the lowest mean squared error for all the

different complexities. We also see in the images that the MSE here is lowest in the figure which has the lowest complexity.

Table 2: Table of the lowest mean squared error of the test data for the terrain data using the different methods.

	OLS	Ridge	Lasso
MSE	35710	12208	8903

We see in Table 2 that for the terrain data the Lasso method gives the lowest mean squared error for the test data. The OLS method gives the highest MSE. Thus the Lasso method seems to be the best method to use when fitting a model to the terrain data. However, we note that all these MSE values are quite high.

Overall, since the model is based on polynomials, it is difficult for the model to explain details of the terrain. The terrain data is not predicted well based on the higher MSE values due to the response varying a lot. Therefore, it is difficult to fit a single function expression over it. To fit the terrain better, we need to introduce a larger complexity with high polynomial degree. The high-degree terms are the most correlated and regularization will shrink these terms.

4 Conclusion

In this project we have looked at three methods of linear regression for the purpose of making assessments by making a prediction model from self-generated data, as well as real terrain data of the norwegian west coast. In order to make assessments we have computed the mean squared error using the resampling techniques K-Fold Cross Validation and Bootstrap. On the self-generated data of the Franke function, we demonstrated that Ridge regression results in the lowest mean squared error, although we were not able to rule out that Lasso regression has potential to perform better although it will make it even more computationally expensive.

In our fitting of the self-generated data we saw some disagreements in the mean squared error calculated using Bootstrap and Cross-Validation. In order to make proper assessments of regression methods it is therefore of interest to explore the validity of resampling techniques like Bootstrap and Cross-Validation as tools for making assessments of regression methods.

We saw from our fitting of the terrain data that of the three methods Lasso is the one that

results in the lowest MSE. However, we also see that these linear regression methods are not very well suited for the terrain data. It would for this data be interesting to apply other methods, perhaps by using a Fourier series as opposed to polynomials to fit the data and see if this will yield better results.

Appendices

A GitHub

The source code and the test results can be found at the address
https://github.com/erikasan/fys-stk-project1/tree/master/project1.

B Analytical Solutions to OLS and Ridge Regression

To find the minima of the cost functions we take the gradient with respect to $\boldsymbol{\beta}$ and set the resulting expression to zero.

$$\frac{\partial}{\partial \boldsymbol{\beta}} C(\boldsymbol{\beta}\,|\mathbf{X}) = \mathbf{0}$$

The components of the gradient of $C^{\text{OLS}}(\boldsymbol{\beta}\,|\mathbf{X})$ are

$$\frac{\partial}{\partial \beta_k} C^{\text{OLS}}(\boldsymbol{\beta}\,|\mathbf{X}) = \frac{1}{n}\sum_{i=0}^{n-1} \frac{\partial}{\partial \beta_k}\left(y_i - \sum_{j=0}^{p-1} X_{ij}\beta_j\right)^2$$

$$= \sum_{i=1}^{N} 2\left(y_i - \sum_{j=1}^{p} X_{ij}\beta_j\right)(-X_{ik}) \propto \sum_{i=1}^{N} X_{ik}y_i - \sum_{i=1}^{N} X_{ik}\sum_{j=1}^{p} X_{ij}\beta_j = 0$$

$$\rightarrow \sum_{i=1}^{N} X_{ki}^T y_i = \sum_{i=1}^{N} X_{ki}^T \sum_{j=1}^{p} X_{ij}\beta_j$$

In matrix notation this equation is

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

which is the condition that $\boldsymbol{\beta}$ must satisfy in order to be a minimum of $C^{\mathrm{OLS}}(\boldsymbol{\beta}\,|\mathbf{X})$. If $\mathbf{X}^T\mathbf{X}$ is not invertible, there are several solutions. If $\mathbf{X}^T\mathbf{X}$ is invertible, the unique solution is given by

$$\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

The components of the gradient of $C^{\mathrm{Ridge}}(\boldsymbol{\beta}\,|\mathbf{X},\lambda)$ are

$$\frac{\partial}{\partial\beta_k}C^{\mathrm{Ridge}}(\boldsymbol{\beta}\,|\mathbf{X},\lambda) = \frac{1}{n}\sum_{i=0}^{n-1}\frac{\partial}{\partial\beta_k}\left(y_i - \sum_{j=0}^{p-1}X_{ij}\beta_j\right)^2 + \lambda\sum_{j=0}^{p-1}\frac{\partial}{\partial\beta_k}\beta_j^2$$

$$\propto \sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1}X_{ij}\beta_j\right)(-X_{ik}) + \lambda\beta_k = \sum_{i=0}^{n-1}(-X_{ik})y_i - \sum_{i=0}^{n-1}\sum_{j=0}^{p-1}(-X_{ik})X_{ij}\beta_j + \lambda\beta_k$$

$$= -\sum_{i=0}^{n-1}X_{ki}^T y_i + \sum_{i=0}^{n-1}X_{ki}^T\sum_{j=0}^{p-1}X_{ij}\beta_j + \lambda\beta_k = 0$$

$$\rightarrow \sum_{i=0}^{n-1}X_{ki}^T\sum_{j=0}^{p-1}X_{ij}\beta_j + \lambda\beta_k = \sum_{i=0}^{n-1}X_{ki}^T y_i$$

In matrix notation this equation is

$$\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$$

If the matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is invertible we get the single solution

$$\hat{\boldsymbol{\beta}}^{\mathrm{Ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

C Variance of the estimators of OLS

The analytical solution to the OLS problem is $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$. For convenience we just write $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$ as $\hat{\boldsymbol{\beta}}$ in the following. We denote the expectation value of $\hat{\boldsymbol{\beta}}$ as $\boldsymbol{\beta}$. The covariance matrix of the components of $\hat{\boldsymbol{\beta}}$ is

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left[(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])^T\right]$$

$$= \mathbb{E}\left[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T\right] = \mathbb{E}\left[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})\left(\hat{\boldsymbol{\beta}}^T - \boldsymbol{\beta}^T\right)\right]$$

$$= \mathbb{E}\left[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T - \hat{\boldsymbol{\beta}}\boldsymbol{\beta}^T - \boldsymbol{\beta}\hat{\boldsymbol{\beta}}^T + \boldsymbol{\beta}\boldsymbol{\beta}^T\right] = \mathbb{E}\left[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T\right] - \mathbb{E}[\hat{\boldsymbol{\beta}}]\boldsymbol{\beta}^T - \boldsymbol{\beta}\mathbb{E}\left[\hat{\boldsymbol{\beta}}^T\right] + \boldsymbol{\beta}\boldsymbol{\beta}^T]$$

$$= \mathbb{E}\left[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T\right] - \boldsymbol{\beta}\boldsymbol{\beta}^T - \boldsymbol{\beta}\boldsymbol{\beta}^T + \boldsymbol{\beta}\boldsymbol{\beta}^T = \mathbb{E}\left[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T\right] - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

$$= \mathbb{E}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}\left\{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}\right\}^T\right] - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

$$= \mathbb{E}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}\mathbf{y}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right] - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

In the last line it has been used that $(\mathbf{ABC})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$ and that $\left\{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right\}^T = \left\{\left(\mathbf{X}^T\mathbf{X}\right)^T\right\}^{-1} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ (the transpose of the inverse of a matrix is equal to the inverse of the transpose of a matrix). Now we have

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

Where the expectation value $\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]$ is

$$\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] = \mathbb{E}\left[(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})^T\right]$$

$$= \mathbb{E}\left[(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})\left(\boldsymbol{\beta}^T\mathbf{X}^T + \boldsymbol{\epsilon}^T\right)\right] = \mathbb{E}\left[\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + \mathbf{X}\boldsymbol{\beta}\boldsymbol{\epsilon}^T + \boldsymbol{\epsilon}\boldsymbol{\beta}^T\mathbf{X}^T + \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right]$$

$$= \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + \mathbf{X}\boldsymbol{\beta}\mathbb{E}\left[\boldsymbol{\epsilon}^T\right] + \mathbb{E}[\boldsymbol{\epsilon}]\boldsymbol{\beta}^T\mathbf{X}^T + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right] = \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right]$$

The last term is the covariance matrix of $\boldsymbol{\epsilon}$ since

$$\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right] = \mathbb{E}\left[(\boldsymbol{\epsilon}-\mathbf{0})(\boldsymbol{\epsilon}-\mathbf{0})^T\right] = \mathbb{E}\left[(\boldsymbol{\epsilon}-\mathbb{E}[\boldsymbol{\epsilon}])(\boldsymbol{\epsilon}-\mathbb{E}[\boldsymbol{\epsilon}])^T\right] = \mathrm{Var}(\boldsymbol{\epsilon})$$

The diagonal elements of $\mathrm{Var}(\boldsymbol{\epsilon})$ are

$$\mathrm{Var}(\boldsymbol{\epsilon})_{ii} = \mathbb{E}\left[(\epsilon_i - \mathbb{E}[\epsilon_i])^2\right] = \mathrm{Var}(\epsilon_i) = \sigma^2$$

The non-diagonal elements of $\mathrm{Var}(\boldsymbol{\epsilon})$ are

$$\mathrm{Var}(\boldsymbol{\epsilon})_{ij} = \mathbb{E}[(\epsilon_i - \mathbb{E}[\epsilon_i])(\epsilon_j - \mathbb{E}[\epsilon_j])]$$

$$= \mathbb{E}[(\epsilon_i - 0)(\epsilon_j - 0)] = \mathbb{E}[\epsilon_i\epsilon_j]$$

$$= \mathbb{E}[\epsilon_i]\mathbb{E}[\epsilon_j] = 0$$

where the last line follows from the assumption that each ϵ_i are independent (but identically distributed) variables. Therefore $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}$ and $\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] = \mathbf{X}\beta\beta^T\mathbf{X}^T + \sigma^2\mathbf{I}$

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{X}\beta\beta^T\mathbf{X}^T + \sigma^2\mathbf{I}\right)\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

$$= \left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-\mathbf{1}}\mathbf{X}^\mathbf{T}\mathbf{X}\beta\beta^T\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} + \sigma^2\left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-\mathbf{1}}\mathbf{X}^\mathbf{T}\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T$$

$$= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2\left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-\mathbf{1}} - \boldsymbol{\beta}\boldsymbol{\beta}^T = \sigma^2\left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-\mathbf{1}}$$

In particular the variance of the components of $\hat{\boldsymbol{\beta}}$ are the diagonal elements of the covariance matrix

$$\mathrm{Var}(\hat{\beta}_i) = \sigma^2 \left[\left(\mathbf{X}^\mathbf{T} \mathbf{X} \right)^{-1} \right]_{ii}$$

D The Bias-Variance Decomposition

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

We will here assume that the y is given by a function with noise, thus we have $\mathbf{y} = f(\mathbf{x}) + \epsilon$. We then insert this into the expression

$$\begin{aligned}
\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] &= \mathbb{E}[(f(\mathbf{x}) + \epsilon - \tilde{\mathbf{y}})^2] \\
&= \mathbb{E}\left[(f(\mathbf{x}) + \epsilon - \tilde{\mathbf{y}} - \mathbb{E}[\tilde{\mathbf{y}}] + \mathbb{E}[\tilde{\mathbf{y}}])^2 \right] \\
&= \mathbb{E}\left[((f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}]) + (\epsilon + \mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}}))^2 \right] \\
&= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2 + 2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\epsilon + \mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}}) + (\epsilon + \mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2] \\
&= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2 + 2\epsilon(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}]) + 2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}}) + (\epsilon + \mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2]
\end{aligned}$$

We have added and subtracted the expectation value of the model here in and have then expanded the expression. If we expand this further we get,

$$= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2 + 2\epsilon(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}]) + 2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}}) + \epsilon^2 + 2\epsilon(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}}) + (\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2]$$

Now we take the expectation value of the terms

$$\begin{aligned}
&= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2] + \mathbb{E}[2\epsilon(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])] \\
&+ \mathbb{E}[2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})] + \mathbb{E}[\epsilon^2] + \mathbb{E}[2\epsilon(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})] + \mathbb{E}[(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2]
\end{aligned}$$

We have chosen normally distributed noise around $\mu = 0$, so we know that the expectation value of the noise is 0. Thus the second and fifth terms disappear.

$$= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2] + \mathbb{E}[2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})^2]$$

The first and last term we can recognise from the cost function,

$$= \frac{1}{n} \sum_{i=0}^{n-1} (f_i - \mathbb{E}[\tilde{y}_i])^2 + \mathbb{E}[2(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])(\mathbb{E}[\tilde{\mathbf{y}}] - \tilde{\mathbf{y}})] + \mathbb{E}[\epsilon^2] + \frac{1}{n} \sum_{i=0}^{n-1} (\mathbb{E}[\tilde{y}_i] - \tilde{y}_i)^2$$

The second term will be zero because we get $\mathbb{E}[\mathbb{E}[\tilde{\mathbf{y}}]] - \mathbb{E}[\tilde{\mathbf{y}}] = \mathbb{E}[\tilde{\mathbf{y}}] - \mathbb{E}[\tilde{\mathbf{y}}] = 0$. The variance for the noise is given as $\sigma^2 = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2$, since we assume that the expectation value of the noise is 0. The expectation value of the noise squared is then the same as the variance. Thus we get,

$$\mathbb{E}[\mathbf{y} - \tilde{\mathbf{y}}] = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - \mathbb{E}[\tilde{y}_i])^2 + \sigma^2 + \frac{1}{n}\sum_{i=0}^{n-1}(\mathbb{E}[\tilde{y}_i] - \tilde{y}_i)^2$$

References

[1] D. C. Lay, S. R. Lay, and J. J. McDonald, *Linear Algebra and Its Applications*, p. 424. Pearson, 2016.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, p. 47. Springer, 2009.

[3] R. Franke and G. Nielson, "Scattered data interpolation and applications: A tutorial and survey," p. 182, 01 1991.