October 6, 2022

Abstract

Abstract

Contents

1 Introduction

In this project we will study the Franke function and real topography data by using regression analysis and resampling methods. These are both two dimensional systems which gives us a way of generalizing our functions to work for both. We will use ordinary least squares, ridge and lasso regression in our study of our data and do comparisons to try finding the optimal model for the different datasets.

2 Method

Our fist step is to set up our data for analysis. We start by implementing the Franke function which is a sum of four exponentials:

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

By using this function we generate our data by using $(n{+}1{\times}n{+}1)$ points in a meshgrid for x and y in the interval [0,1]. This gives us a function value of $(n+1\times n+1)$ to which we add a stochastic noise $\epsilon \sim N(0,\sigma^2)$ for a chosen standard deviation σ. This leaves us with our data described by the franke function plus some noise ϵ:

$$\boldsymbol{z} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

This can also be assumed of some real data, for the assumtion of the existence of a continous function f and normal distributed error ϵ.

The ordinary least squares gives us an aproximation to the above equation where we minimize $(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2$ to give us the matrix equation:

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

for a chosen design matrix $\boldsymbol{X}$ of some degree n.

This approximation gives us a new way of describing our dataset in terms of $\tilde{\boldsymbol{z}}$ instead of $f(\boldsymbol{x})$.

$$\boldsymbol{z} = \sum_{j=0}^{n-1} \beta_j x_i^j + \epsilon_i$$

This is the same as the following matrix equation:

$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} = \tilde{\boldsymbol{z}} + \boldsymbol{\epsilon}$$

This means that the elemnt i is given by:

$$z_i = \epsilon_i + \sum_j x_{ij}\beta_j$$

and the expectation value of the element i in $\boldsymbol{z}$:

$$E(z_i) = E(\epsilon_i + \sum_j x_{ij}\beta_j)$$

Since we already know or have assumed that $\boldsymbol{\epsilon}$ is normal distibuted with the expectation value 0 $E(\epsilon_i) = 0$, this gives us:

$$E(z_i) = E(\epsilon_i) + E(\sum_j x_{ij}\beta_j) = \sum_j E(x_{ij}\beta_j)$$

x_{ij} and β_j are values and not distrubutions which means they have themselves as expectation values:

$$E(z_i) = \sum_j x_{ij}\beta_j = \boldsymbol{X_{i*}}\boldsymbol{\beta}$$

To find the variance we again recognize that $\boldsymbol{X}\boldsymbol{\beta}$ follow no distribution which leaves us with the variance of $\boldsymbol{z}$ equaling the variance of $\boldsymbol{\epsilon}$ which is given as σ^2:

$$\mathrm{Var}(\boldsymbol{z}) = \mathrm{Var}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) = \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2$$

$\tilde{\boldsymbol{z}}$ is defined through the minimization of the mean square error $(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2$ which for $\tilde{\boldsymbol{z}} = \boldsymbol{X}\boldsymbol{\beta}$ translates to the minimization of the cost function:

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

Where we take the derivative with respect to $\boldsymbol{\beta}$ and solve where the derivative is 0 to find the minimum:

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

$$\boldsymbol{X}^T\boldsymbol{z} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}$$

We assume that $\boldsymbol{X}^T\boldsymbol{X}$ is invertible which gives us the the optimal $\boldsymbol{\beta}$:

$$\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}$$

We can now find the expectation value for the optimal $\tilde{\boldsymbol{\beta}}$:

$$\begin{aligned} E(\tilde{\boldsymbol{\beta}}) &= E[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}] = E[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T E[(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta} \end{aligned}$$

Here we see that the expectation value of our optimal parameter is the parameter $\boldsymbol{\beta}$.

We now find the variance of the optimal parameter:

$$\begin{aligned} \mathrm{Var}(\tilde{\boldsymbol{\beta}}) &= \mathrm{Var}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\ &= \mathrm{Var}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\epsilon}] \end{aligned}$$

For a matrix A we have $\mathrm{Var}(\boldsymbol{A}\boldsymbol{X}+b) = \boldsymbol{A}\mathrm{Var}(\boldsymbol{X})\boldsymbol{A}^T$ which gives us:

$$\begin{aligned} \mathrm{Var}(\tilde{\boldsymbol{\beta}}) &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathrm{Var}[\boldsymbol{\epsilon}]((\boldsymbol{X}^T\boldsymbol{X})^{-1})^T\boldsymbol{X} \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathrm{Var}[\boldsymbol{\epsilon}](\boldsymbol{X}^T)^{-1}\boldsymbol{X}^{-1}\boldsymbol{X} \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\mathrm{Var}[\boldsymbol{\epsilon}] \end{aligned}$$

This we can use to compute the optimal β and in turn give us a OLS prediction for our real function f.

2.1 Splitting and scaling of data

A central part of regression analysis is to scale and split our data to avoid both overfitting and avoid the MSE to contain the intercept. In our case of the Franke function we have $x, y \in [0, 1]$ and follwing z data on the same order which may indicate that a scaling is unessecary. This expectation comes from that our data does not include any extreme values of any other order that may greatly negativly incluence our predictors. In other words scaling of data is normaly what we do to get the data as equal to each other as our Franke data already is. To test if this

expectation is valid, we test this and compare our non-scaled data to mean scaled data where we have subtracted the mean.

To avoid overfitting we perform a 80-20 percent split into train and test data. We use the train data to train the model or in other words find the β parameters. To predict we use the trained model together with the test data which leaves us with a more generalized prediction not only working for the data we have at hand, but also other data coming from the same source. This can be explained by for example a model trained so well with a design matrix $\boldsymbol{X}$ of an infinite degree that its prediction follows every noisy outlier. This woulds match the data at hand very well and give us an extremly low MSE, but give us high MSE for new data which were not used to train the model. To test this we look at both MSE and R squared for predictions for $\boldsymbol{X}$ of different degrees using only train data compared to predictions made using both train and test data.

2.2 Mean squared error and R squared

For different design matrices X with different polynomial degrees we can calculate the mean squared error and R squared. This is done using the following equations where we define z as our test data containing $n+1$ points.

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

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

For the mean value:

$$\bar{z} = \frac{1}{n}\sum_{i=0}^{n} z_i$$

2.3 Bias-variance trade-off

To study the bias-variance trade-off we first implement the bootstrap resampling technique. This is done by resampling our train data n_B times with replacement. That means we gain n_B new samples that may be unique but still only contain values from the original sample. This in turn is used to compute our predictions leaving us with a total of n_B different predictions. This can be used to study the bias and

variance of the predictions.

By looking back at the cost function which is minimized in order to find our optimal parameters $\boldsymbol{\beta}$ we can derive an expression for the bias and variance.

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

By the definition of our data $z = f + \epsilon$ we rewrite the costfunction:

$$\begin{aligned}
\mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \mathbb{E}[(f + \epsilon - \tilde{z} + \mathbb{E}[\tilde{z}] - \mathbb{E}[\tilde{z}])^2] \\
&= \mathbb{E}[f^2 + \epsilon^2 + \tilde{z}^2 + 2\,\mathbb{E}[\tilde{z}]^2 \\
&+ 2f\epsilon + 2f\tilde{z} + 2f\,\mathbb{E}[\tilde{z}] - 2f\,\mathbb{E}[\tilde{z}] \\
&+ 2\epsilon\tilde{z} + 2\epsilon\,\mathbb{E}[\tilde{z}] - 2\epsilon\,\mathbb{E}[\tilde{z}] \\
&+ 2\tilde{z}\,\mathbb{E}[\tilde{z}] - 2\tilde{z}\,\mathbb{E}[\tilde{z}] - 2\,\mathbb{E}[\tilde{z}]^2] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})^2] \\
&+ 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[\epsilon] + 2\,\mathbb{E}[\epsilon]\,\mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})] \\
&+ 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})]
\end{aligned}$$

By using that $\epsilon \sim N(0, 1)$ we have $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}[\epsilon] = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2 = \mathbb{E}[\epsilon^2] = \sigma^2$ from this we get:

$$\begin{aligned}
\mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 + 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[\mathbb{E}[\tilde{z}] - \tilde{z}] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathrm{var}[\tilde{f}] + \sigma^2
\end{aligned}$$

We see that the first term is $(\mathrm{Bias}[\tilde{f}])^2$ but we are left with a problem to compute it because f is unknown. To be able to find the bias we look at the computeable $E[(y - \mathbb{E}[\tilde{y}])^2]$ and the relation $z = f + \epsilon$:

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

We recognize once again that we can remove terms that have ϵ in them from $\mathbb{E}[\epsilon] = 0$

giving us:

$$\mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] = E[(y - \mathbb{E}[\tilde{y}])^2] - \sigma^2$$

And we end up with the the realtion:

$$\begin{aligned}\mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \text{Bias}[\tilde{f}] + \text{var}\tilde{f} + \sigma^2 \\ &= (\text{Bias}[\tilde{y}] - \sigma^2) + \text{var}[\tilde{f}] + \sigma^2\end{aligned}$$

In other words we compute the correct bias for our models by subtracting the variance of our noise.

2.4 Cross validation

To futher investigate our model we implement the cross validation resampling method. Togther with the already implemented bootstrap method we now have a possibility to optimize our predictions by chosing a design matrix that reduces our models mean squared errors. The cross validation method takes in data and splits it up in k equaly sized sets where one set is used as test data while the rest is used as train data. For every fold a mean squared error is then computed before the model reruns with another set as test data. This process happens over and over until all k sets have been used as test data. This gives a total of k folds. Since our data is ordered and not chaotic we shuffle it before the splits are performed. We can see why this is neccesary in the k-fold visualization in figure 1 where we would without a shuffle have x, y an z not representative of the whole dataset but rather small intervals of values. We can imagine how not shuffling the data would have consequences on a extreme case where the first split of the first k-fold in the figure below only contained values around 0 while the rest contained values around 1. That would leave us with high computed mean squared erros for especially some of the folds which would in turn give us bad predictions for the MSE.

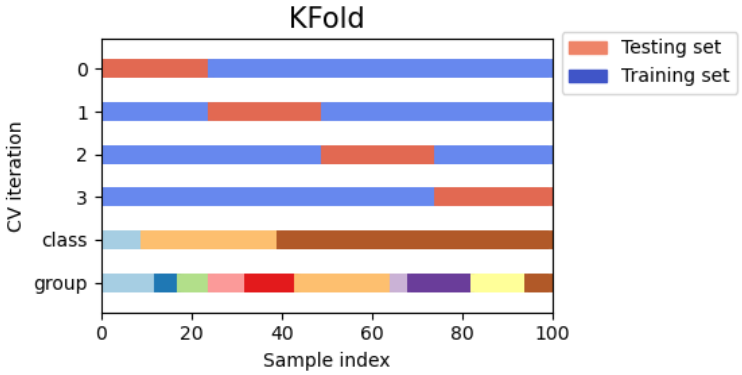


Figure 1: A visualization of the splits of the data performed in cross validation. Here we see 4 k-folds which gives a total of 4 splits and iterations.
https://scikit-learn.org/stable/modules/cross_validation.html

2.5 Study of λ dependence for ridge and lasso regression

To get the best possible predictions of our datasets we also implement lasso and ridge regression. These two methods are unlike OLS, dependent on a regularization parameter λ where we have to compute which value that optimizes these two models. Similar to OLS we perform cross validation to find which degree of design matrix reduces the mean squared error of our predictions, only that we for ridge and lasso find the best λ at the same time. This is because we for one degree would have one optimal λ and for another degree have another. We therefore compute one cross validation for different values of λ parameters for each degree. The degree and λ giving the smallest MSE is then used for futher predictions

2.6 Study of topography data

After studying the franke function with added noise we look at real topography data from the mountains close to Stavanger in Norway. We perform the same analysis as we did with the Franke function except that we now assume that our data already contain some noise $\epsilon \sim N(0, \sigma^2)$ with an unknown standard deviation σ. We first standard scale our data by subtracting the mean and dividing by the standard deviation computed from the data:

$$z_{scaled} = \frac{z - \bar{z}}{\hat{\sigma}_z}$$

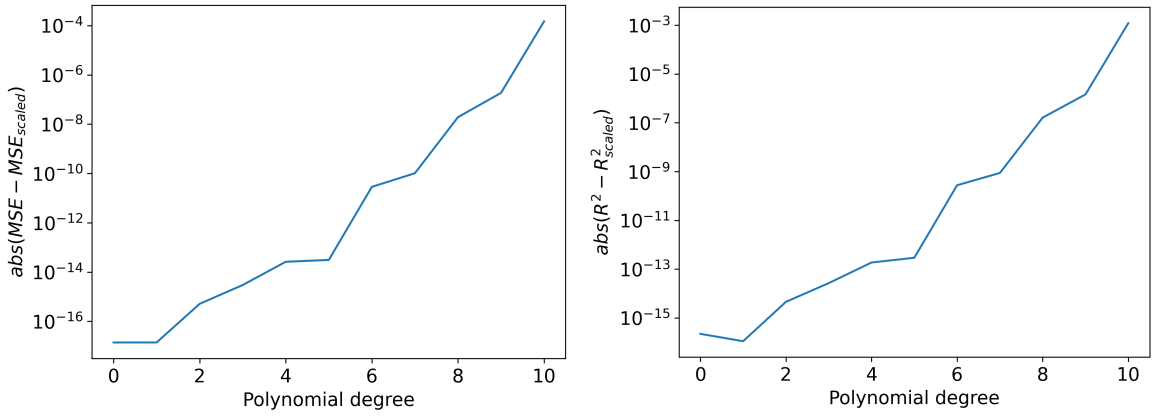
Our regression analysis will then be used to try to compute how the topography at the location actualy looks. Since the imported data includes 3601×1801 datapoints which would require more powerful hardware or atleast take a significant time to analyze, we choose an interesting location of indexes 100 to 141 in both directions and slice the data for every second index. That means we are left with input data of lower resoulution, but at the same time it enables us to do an analysis over a greater area.

3 Results

3.1 Franke funtion

3.1.1 OLS scaling of data

We first look if there is a need of scaling our data or not. As seen in figure 2 we have plotted the absoulute difference between the computed MSE and R^2 for different polynomial degrees. We see that the difference is so small especially for the 3 we see them both plotted for design matrices up to a degree of 5 for $n = 30$ steps and $\sigma = 0.2$.



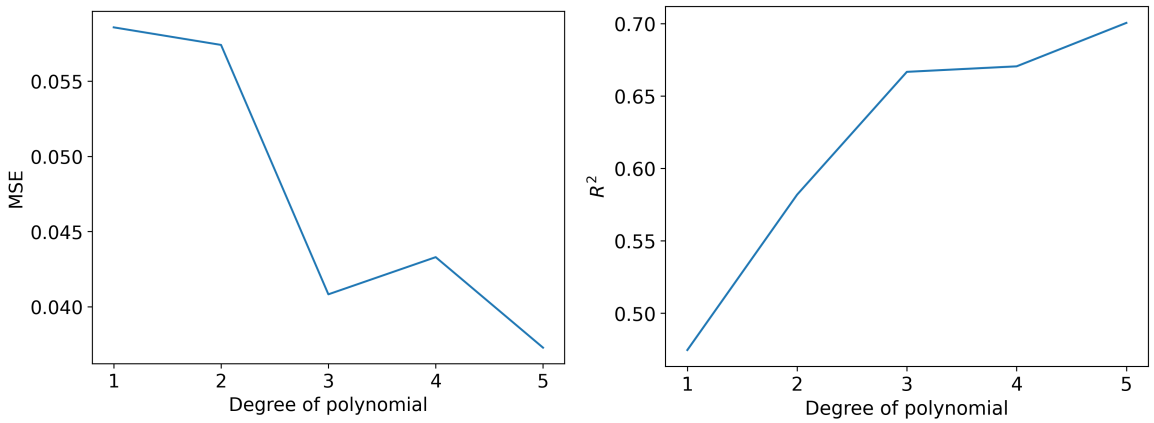
(a) MSE (b) R^2

Figure 2: Comparison of MSE and R^2 between scaled and non-scaled data

We see that the difference between the scaled and non-scaled data are so small especially for lower polynomial degrees which indicate that scaling of our data is unnecesary.

3.1.2 OLS MSE and R^2

We continue with our non-scaled data and look at how the mean square error and R squared behave for design matrices of different degrees. Underneeth in figure 3 we see them both plotted for design matrices up to a degree of 5 for $n = 30$ steps and $\sigma = 0.2$.

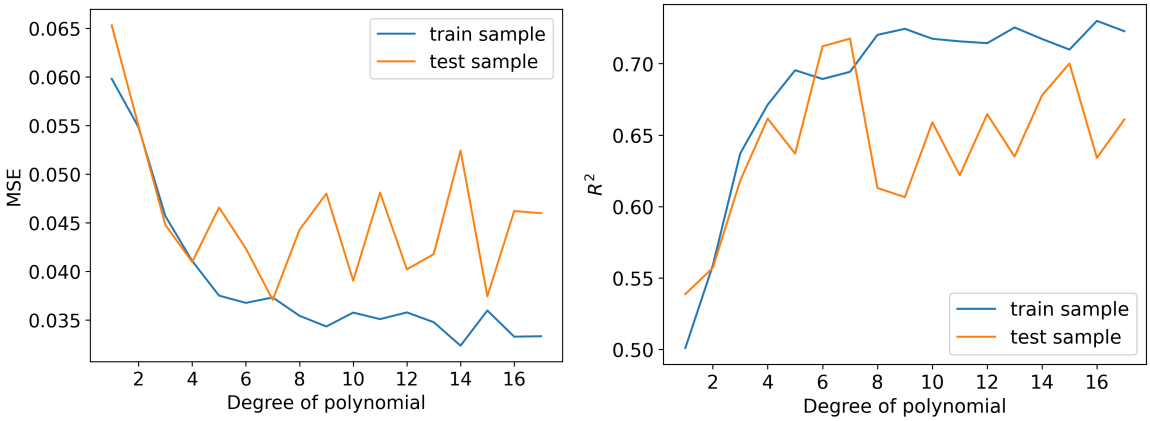


(a) MSE (b) R^2

Figure 3: plots of MSE and R^2 for the franke function for an error $\epsilon \sim N(0, \sigma^2)$ with $\sigma = 0.2$ and 31 points in each direction

Above in figure 3 we see a clear reduction in MSE for an increase in polynomial degree of the design matrix. For R^2 we see increasing values for higher degrees, with degree 5 giving us the lowest MSE and highest R^2 indicating degree 5 being the most optimal degree.

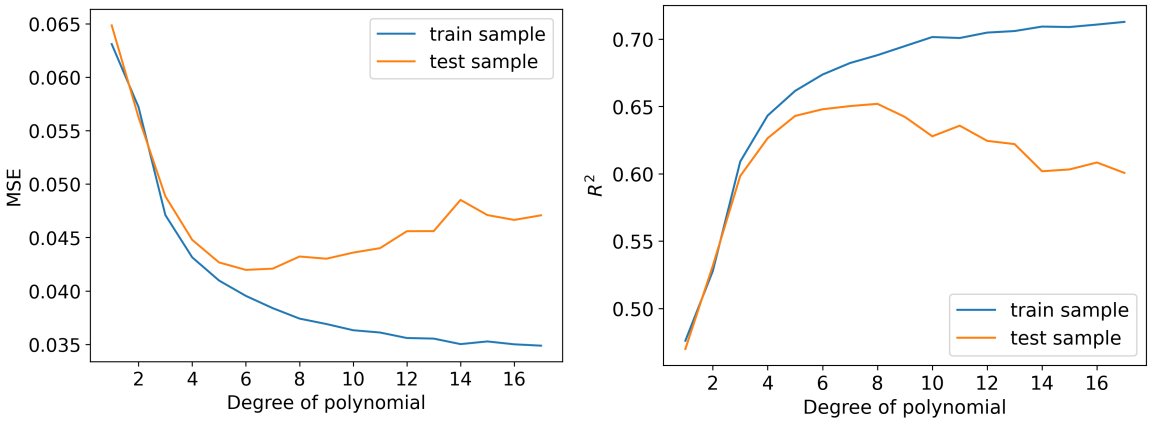
We continue to look into MSE and R^2 but now comparing the use of train or test data in the prediction of z. We continue using $n = 30$ steps and $\sigma = 0.2$. In figure 4 we see a single run and the following MSE and R^2 while we in figure 5 see a plot of a resample of the noise data giving us 100 unique datasets to take the mean of.



(a) MSE (b) R^2

Figure 4: MSE and R^2 computed from both the train and test sample

Above we see that both the MSE and R^2 from the test sample seem to start deviating from the train sample at degrees higher than 5 which indicates overfitting for greater degrees.



(a) MSE (b) R^2

Figure 5: An average of MSE and R^2 computed from both the train and test sample over 100 unique samples

Above in figure 5 we get a smoother plot showing the smallest MSE obtained from a degree of 6 indicating this being the best polynomial degree for OLS regression of the franke function. The R^2 show the same tendency but looks like having the greatest values at a degree of 8.

3.1.3 OLS variation of β paramers

To see how the β parameters variate from different choices of degrees we plot them for a degree up to 5 indicated in figure 6 below:

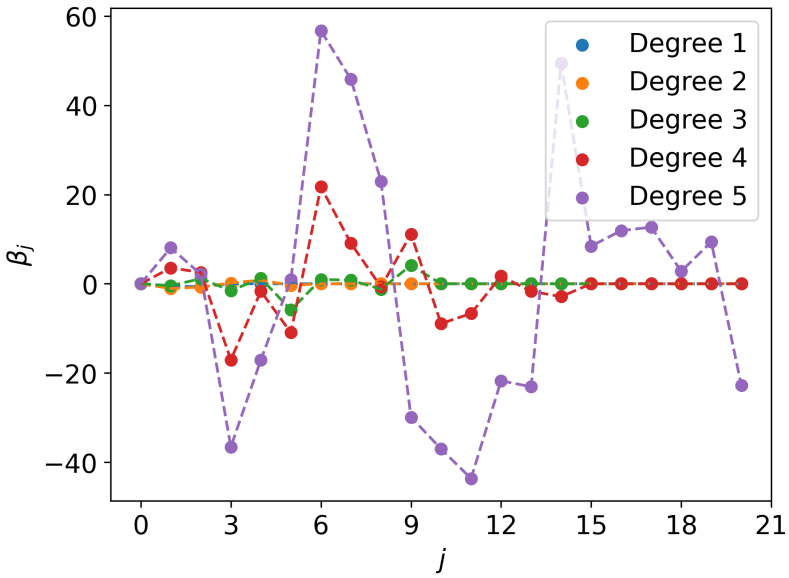


Figure 6

We see an increase in distance between the β parameteres for an increasing polynomial degree. When we at the same time look at table 1 we see that the parameters for a degree of 5 come with great variance. This is especially true for parameters of index in the interval [6,14] which we also see in figure 6 have larger values than the rest. This may be an indication of overfitting, but looking back at our resampled train test MSE plot in figure 9a we see that we for a degree of 5 have lower MSE than larger polynomial degrees.

Table 1: Variance of the β parameters for test and train data for a design matrix of polynomial degree 5. Calculated by $Var(\boldsymbol{\beta}) = diag(\sigma^2(X^TX)^{-1})$

$\boldsymbol{\beta}$	**Test data**	**Train data**
β_0	0.04	0.01
β_1	5.22	1.31
β_2	6.00	1.27
β_3	139.88	31.56
β_4	90.01	19.04
β_5	149.50	31.37
β_6	723.58	162.55
β_7	364.34	90.68
β_8	555.61	83.97
β_9	787.29	163.88
β_{10}	794.84	179.33
β_{11}	521.51	104.67
β_{12}	424.30	88.62
β_{13}	625.04	98.00
β_{14}	880.18	180.91
β_{15}	122.43	27.74
β_{16}	102.30	20.80
β_{17}	80.15	19.80
β_{18}	108.00	19.14
β_{19}	105.44	20.14
β_{20}	134.90	27.84

3.1.4 OLS bias variance tradeoff

To futher analyze our models dependence on polynomial degree we have a plot of the bias variance tradeoff in figure 7 for a degree up to 15 using $n = 22$ steps and a noise standard deviation of $\sigma = 0.2$.

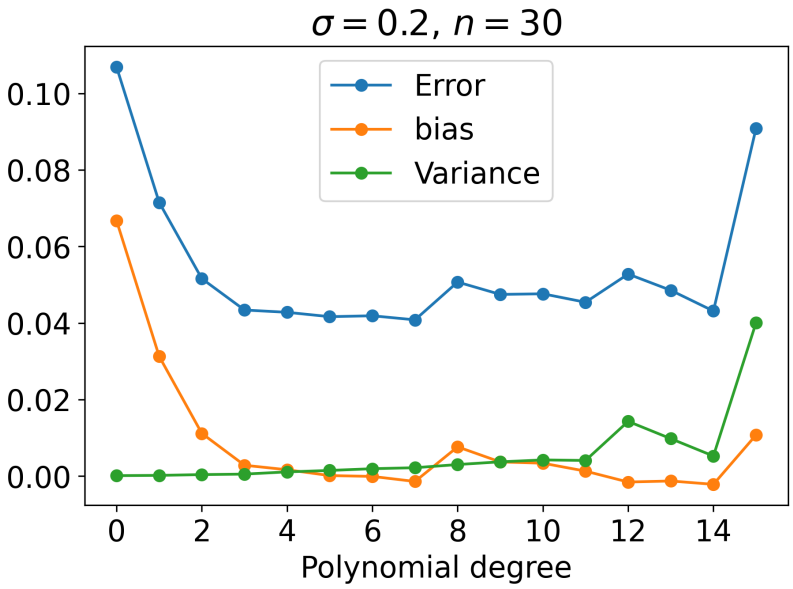


Figure 7: The bias variance tradeoff for the Franke function using 100 bootstrap iteration together with $n = 22$ steps and a noise standard deviation of $\sigma = 0.2$

We se above in figure 7 areas of both high variance and low bias and low variance and relative high bias. We also see that the area of lowest MSE for a degree of 6 is where both the bias and variance together is at their lowest. A futher increase in polynomial degree gives mostly a variance dominated error contribution while we at lower polynomial degrees see that the bias is what contributes the most to the error. When we look at figure 8 we see another trend especially for higher polynomial degrees. Here the variance stays low which may together with the low mse indicate that we dont as easy overfit our data when we have more of it at hand.

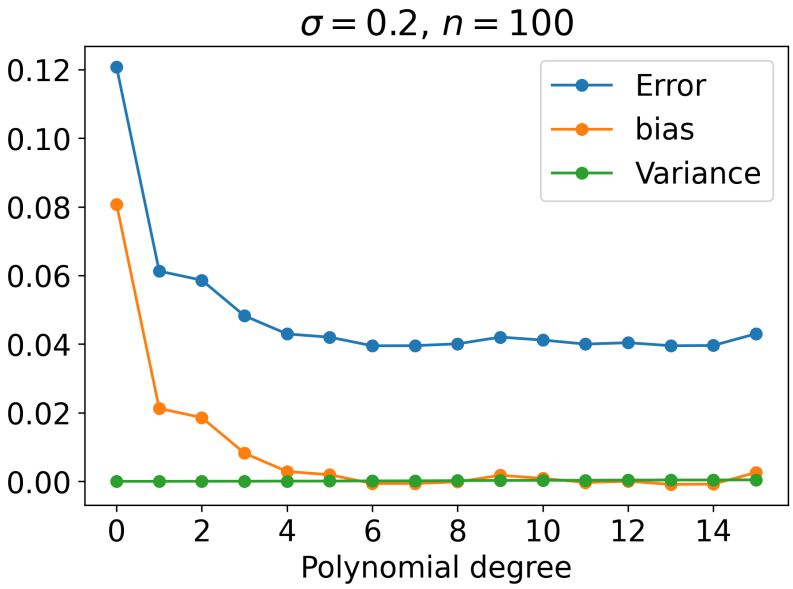
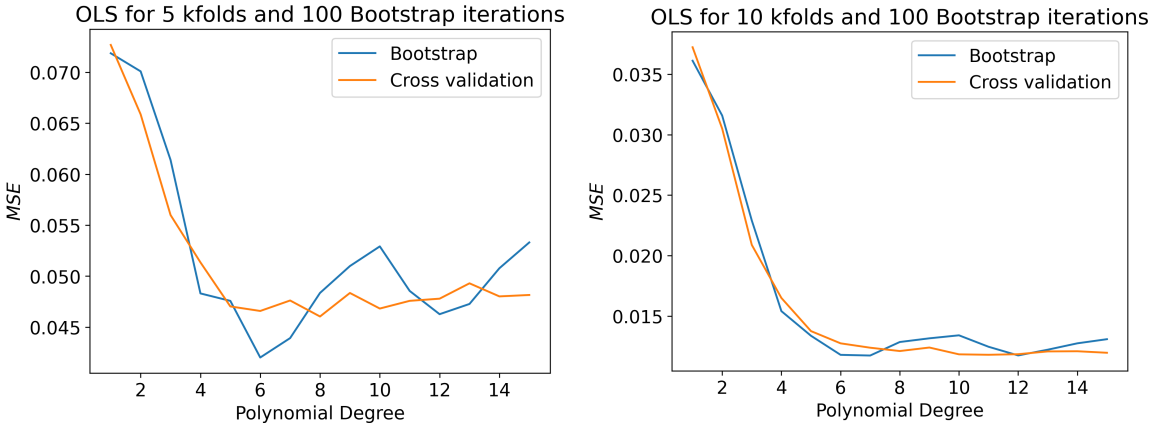


Figure 8: The bias variance tradeoff for the Franke function using 100 bootstrap iteration together with $n = 100$ steps and a noise standard deviation of $\sigma = 0.2$

3.1.5 OLS cross validation

From the implementation of cross validation we get the following comparison plot in figure 9 for data with $n = 30$ steps and $\sigma = 0.2$ for 100 bootstrap iterations compared to cross validation of both 5 and 10 k-folds:



(a) 5 k-folds

(b) 10 k-folds

Figure 9: A comparison between bootstrap and cross validation for datasets with $n = 30$ steps, noise standard deviation $\sigma = 0.2$ and 100 bootstrap iterations

We see different results for bootstrap and cross validation for both 5 and 10 k-folds. The difference between the k-folds are quite small, but the difference from the

bootstrap is for some polynomial degrees higher than others. Underneeth in figure 10 we see a comparison for the same amount of data only with an error standard deviation of $\sigma = 0.1$ instead of 0.2:

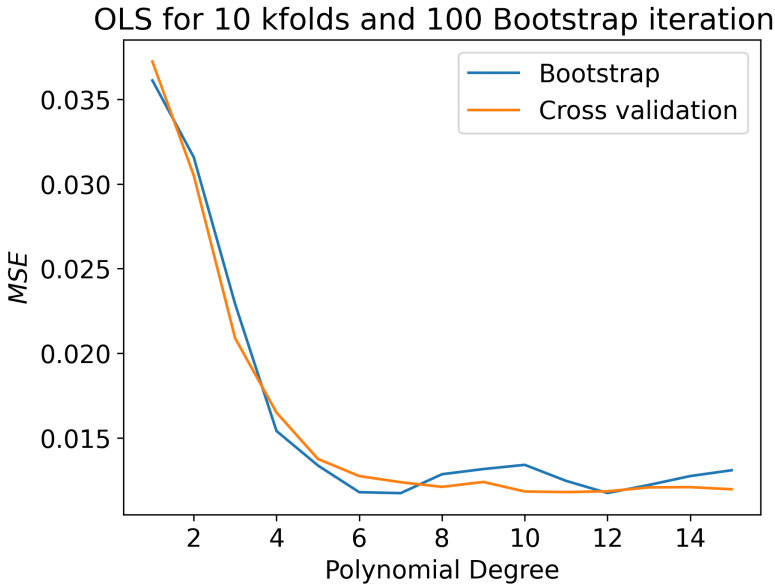


Figure 10: A comparison between bootstrap and cross validation for datasets with $n = 30$ steps, noise standard deviation $\sigma = 0.1$ and 100 bootstrap iterations

We see above in figure 10 that the bootstrap gives more similar results to the cross validation method for a lower noise standard deviation. At the same time we see the same trend for the bootstrap with lower MSE values around a degree of 6 and 12 while it still gives higher MSE for around a degree of 10 and above 13.

We continue using the cross validation method to compute for which degree we get the lowest mean squared errors. As we see under in figure 11

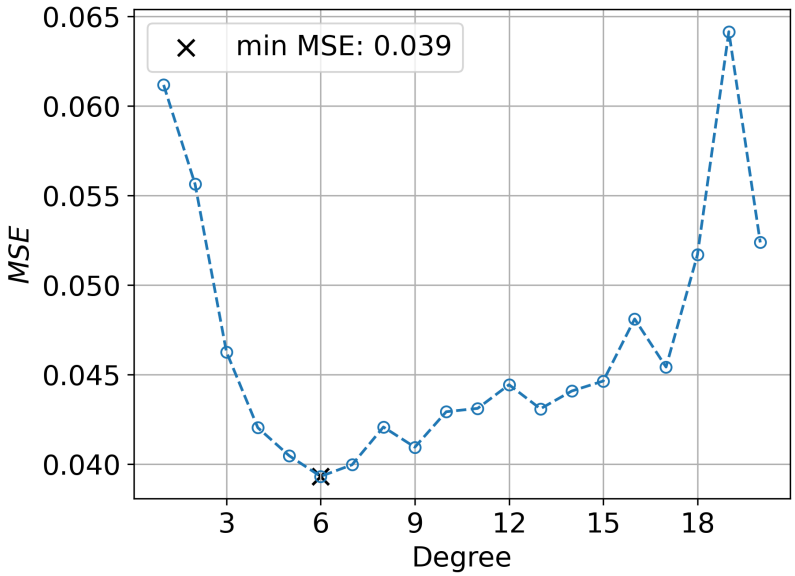
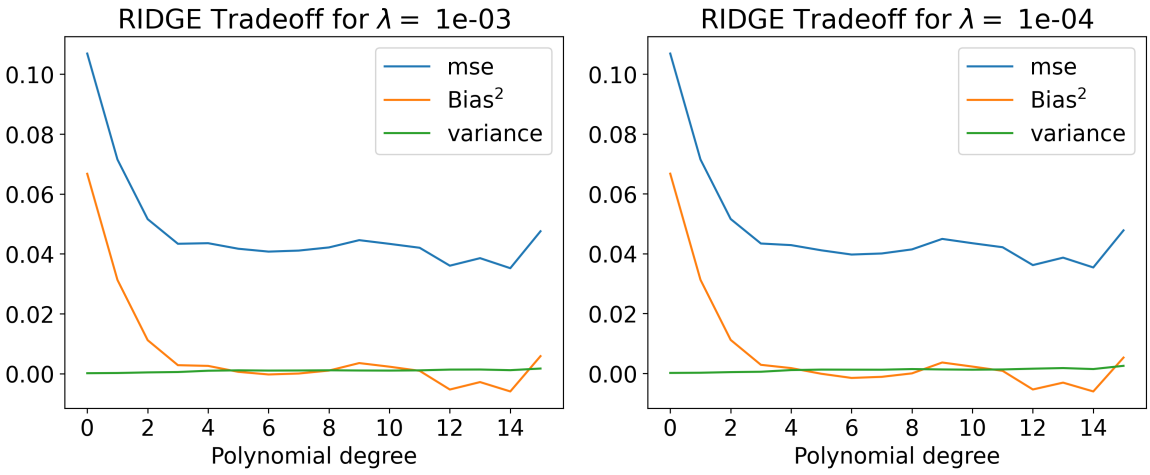


Figure 11: Cross validation to find which polynomial degree gives the lowest MSE for data with $n = 30$ steps and noise standard deviation $\sigma = 0.2$

Above in figure 11 we see similar to figure 9 That we for a degree of 6 minimize the MSE for OLS regression. This means that if we want to compute the best possible prediction $\tilde{z}$ we should use a design matrix of degree 6.

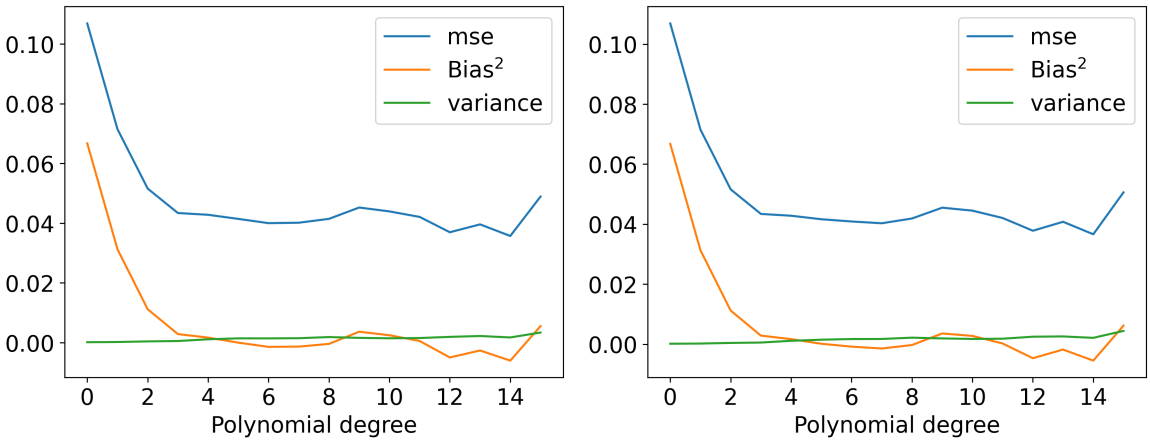
3.1.6 Ridge and Lasso bias variance tradeoff

In the same way as for ordinary least squares we look at the bias variance tradeoff, only that we now have another parameter λ that influences the tradeoff. Under in figure **??** we see the tradeoff for different choices of λ



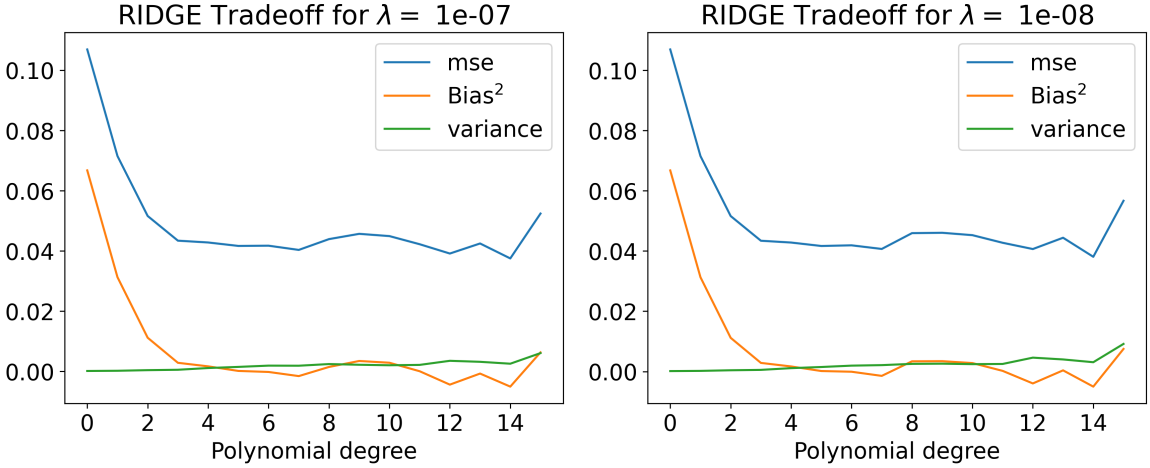
(a)

(b)



(c)

(d)



(e)

(f)

Figure 12

4 Discussion

5 Conclusion

6