University of Oslo

FYS-STK3155

Project 1

Brage Wiseth	bragewi@ifi.uio.no
Felix Cameren Heyerdahl	felixch@ifi.uio.no
Eirik Bjørnson Jahr	eiribja@ifi.uio.no

October 15, 2023

https://github.com/bragewiseth/MachineLearningProjects

Contents

https://github.com/bragewiseth/MachineLearningProjects

Abstract

In this paper, we delve into the realm of machine learning model optimization and evaluation. Our study encompasses various regression techniques, including Ordinary Least Squares (OLS), Ridge, and Lasso regression, to analyze their effectiveness in handling simple and more complex datasets. Additionally, we employ bootstrap resampling and cross-validation methodologies to rigorously assess model performance and enhance generalization. A significant portion of our investigation is dedicated to understanding the delicate balance between bias and variance. We explore how regularization methods like Ridge and Lasso impact bias-variance trade-offs, offering insights into the stability and predictive power of these models. Furthermore, we provide empirical evidence of the benefits of cross-validation and bootstrap techniques in mitigating overfitting and improving model robustness. We found that plain OLS worked best for more simple data, while ridge performed better for more complex data. We found that bootstrap and cross validation helped us get better predictions, and that scaling the data was crucial for regularization techniques. . Additionally we verify and compare our findings with well established theory and libraris such as SKLearn.

Keywords: Linear Regression, Scaling, Bias & Variance

1. Introduction

Machine learning has emerged as a powerful tool in data analysis, providing the ability to uncover complex patterns and relationships in diverse datasets. But, at its core, machine learning is all about finding functions that capture the underlying structure of the data. The use of machine learning algorithms to approximate functions is the essence of this paper.

Our motivation for this research lies in the exploration of machine learning techniques to approximate the terrain on our planet, which can perhaps be described by such a function. Earth's terrain exhibits peaks and troughs, hills and valleys, much like some polynomial functions. Fortunately, we can employ standard linear regression techniques to approximate polynomials, but the terrain presents its own set of challenges. Firstly, the terrain's true underlying function may not be a polynomial at all, and its complexity may vary significantly from one location to another. Secondly, our landscape is teeming with small, intricate details. Some regions are characterized by flat and smooth surfaces, while others are marked by rough and uneven terrain. Focusing too much on these minute details can lead to model overfitting, making it crucial to strike a careful balance between model complexity and generalization. In this context, regularization and resampling techniques, including Ridge and Lasso regression with bootstrap and cross validation, have proven indispensable. By introducing regularization and resampling, we aim to find the sweet spot between bias and Variance. And getting the best predictions we can with our assumptions.
To embark on this exploration, we will begin with a simpler case: ”Franke's function.” which mimics our real terrain data. This function serves as a foundational starting point, allowing us to assess our model's performance in a controlled environment before venturing into the complexity of real-world terrain data. Through this gradual progression, we provide ourselves wih a framework that can be applied to more complex and varied real-world terrain datasets.

Data: We begin by introducing the dataset used for our analysis, highlighting data collection and preprocessing procedures. Understanding the characteristics of the terrain data is fundamental to our modeling endeavor.
Methods and Scaling: Next, we delve into the methodology, encompassing the implementation of polynomial regression models and the application of regularization techniques such as Ridge and Lasso. Additionally, we will discuss the importance of proper scaling for model stability and convergence.
Results: In this section, we will present the outcomes of our experiments, showcasing the performance of different models and regularization techniques. Through empirical evidence, we aim to provide insights into the effectiveness of our approach.
Bias-Variance Trade-off: A significant portion of our study will revolve around the critical concept of bias and variance. We'll explore how regularization methods influence this trade-off and delve into the fine balance between model complexity and generalization.
Conclusion: Finally, we will summarize the key findings and their implications for terrain modeling with machine learning. Our conclusion will underscore the importance of regularization in achieving accurate representations of complex terrains and provide a perspective on future research directions.

code for generating all figures and data can be found at `/MachineLearningProjects/project1/src`

2. Data

The data we will be using consists of 100 points generated using the Franke function. The Franke function is a weighted sum of four exponentials given by

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

Figure 1 shows a graphical representation of the Franke's function
We will add some noise to the data to better simulate real world data. The noise is sampled from a normal distribution with mean 0 and variance $\sigma^2 = 1$. The data is generated by sampling x and y from a uniform distribution on the interval $[0, 1]$. In Figure 2 you can see a representation of our data. To apply our findings with this toy dataset to real-world data, we will also use a real topographic dataset from Norway see Figure 3 and Figure 4. However, due to limited computational resources and the time it takes to run our program, we only sample 40000 - 50000 points from about 3 million points. For both datasets we split the data is into training and testing sets, with the testing set comprising twenty percent. For some cases when running resampling techniques we also split the data into training, validation and testing sets.

As we hinted at in the introduction, we will try to fit a polynomial to both datasets. That means that our input data will be the x and y values as well as higher order terms x^2, y^2, xy, x^3, y^3, x^2y, xy^2 and so on. We have chosen to not include the first term of the polynomial, which is just 1. This is because we will be scaling our data, more on this in Section 3.3.

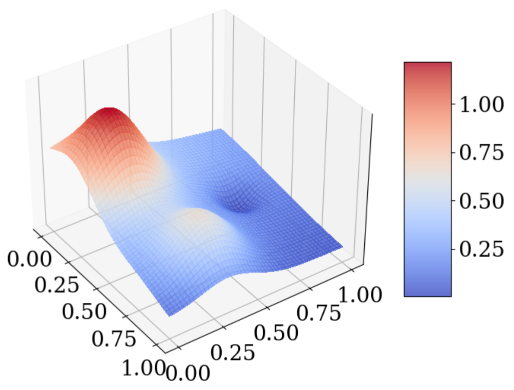

Figure 1: True Function

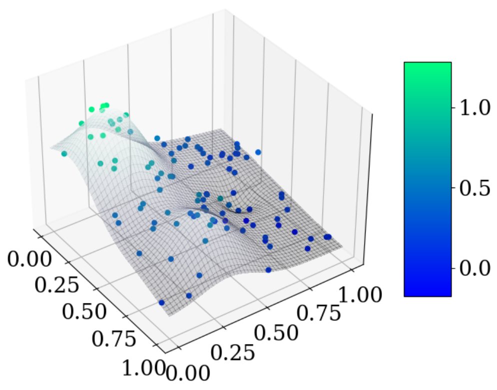

Figure 2: Our Synthetic Data

3. linear regression models

Linear models assume that the relationship between the input variables (predictors or features) and the output variable (response or target) is linear. The predictions of the model is a linear combination of the input variables with some coefficients. The coefficients are estimated from the training data using a method called least squares. The least squares method is used to find the coefficients that minimizes the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. The residual sum of squares is given by

$$RSS(\beta) = \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2$$

where y_i is the observed response for observation i, $\tilde{y}_i$ is the predicted response for observation i

$$\bar{y} = \frac{1}{n} \sum_{i=0}^{n-1} y_i$$

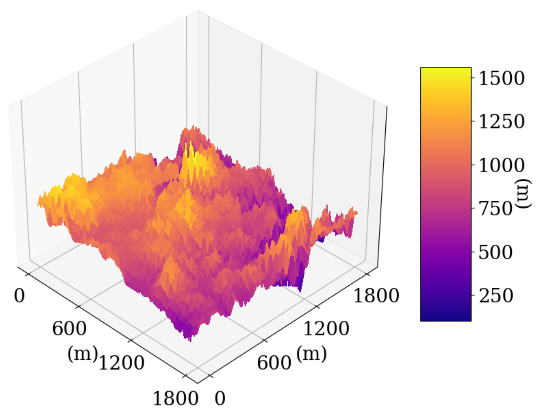

Figure 3: Real data 3D

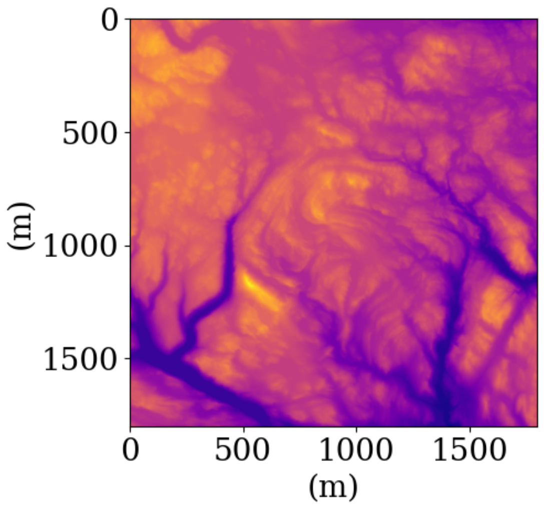

Figure 4: Real data top down

We can also take the mean of the residual sum of squares, which is called the mean squared error (MSE). This is what we will use to evaluate our models. The MSE is given by

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

We wil also use the R^2 score to evaluate our models. The R^2 score is a statistical measure of how close the data are to the fitted regression line. The R^2 score is given by

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

The R2 score is a number between 0 and 1, where 1 indicates that the model fits the data perfectly. A negative R2 score indicates that the model is worse than just predicting the mean of the data.

The goal is to find the coefficients $\boldsymbol{\beta}$ that minimizes the MSE. We can write the predictions $\tilde{\mathbf{y}}$ as

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

Wich is a compact way of writing

$$\tilde{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_{p-1} x_{i,p-1}$$

where p is the number of predictors.

3.1 Ordinary Least Squares (OLS)

By minimizing the MSE, we find the coefficients that minimizes the residual sum of squares. This is done by taking the derivative of the MSE with respect to the coefficients β_j and setting it equal to zero. This gives us the optimal parameters

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

The derivation of this can be found in the appendix A.1

The *Design Matrix* $\mathbf{X}$ is a matrix of our input data, the columns of $\mathbf{X}$ are the features of our data and the rows are the observations. For our case the columns are the polynomial terms. [1]
We can then use these parameters to predict the response $\tilde{\mathbf{y}}$ for a new set of predictors $\mathbf{X}$. The predictions is as previously mentioned a linear combination of the predictors and the coefficients.

3.2 Ridge & Lasso Regression

We can add add regularization to the linear regression model to help keep our parameters in check. This is done by adding a penalty term to the cost function. The penalty term is a function of the coefficients β_j. The penalty term is called the regularization term, and the parameter λ is called the regularization parameter. The regularization parameter determines how much the coefficients are penalized. The cost function for Ridge regression is given by

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 + \lambda\sum_{j=0}^{p-1}\beta_j^2 = \frac{1}{n}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_2^2$$

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

Lasso regression is similar to Ridge regression, with the difference being that the penalty term is the sum of the absolute values of the coefficients β_j, also called the L1 norm. The cost function for Lasso regression is given by

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 + \lambda\sum_{j=0}^{p-1}|\beta_j| = \frac{1}{n}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1$$

We won't be utilizing our custom Lasso implementation and derivation; instead, we will opt for the Lasso implementation provided by the Scikit-Learn library.

3.3 Scaling

Consider Table 1, Table 2 and Table 3. At first glance, there isn't a big difference between the unscaled data and the scaled data. This is because the original data was already close to having a mean close to zero and not being spread out too much. So, you might think that

1. the first term of a polynomial is just 1, however we have chosen not to include this. more on this in Section 3.3 about scaling

Table 1: Unscaled design matrix fitting one-dimensional polynomial of degree 5

1.	0.	0.	0.	0.	0.
1.	0.25	0.0625	0.01562	0.00391	0.00098
1.	0.5	0.25	0.125	0.0625	0.03125
1.	0.75	0.5625	0.42188	0.31641	0.2373
1.	1.	1.	1.	1.	1.

Table 2: Scaled design matrix fitting one-dimensional polynomial of degree 5

0.	-1.41421	-1.0171	-0.83189	-0.728	-0.66226
0.	-0.70711	-0.84758	-0.7903	-0.71772	-0.65971
0.	0.	-0.33903	-0.49913	-0.56348	-0.58075
0.	0.70711	0.50855	0.29116	0.10488	-0.0433
0.	1.41421	1.69516	1.83016	1.90431	1.94603

Table 3: MSE for scaled and unscaled data

mse for ols on unscaled data:	0.010349396022903145
mse for ols on scaled data:	0.010349396024145656
mse for ridge on unscaled data:	0.02106077418650843
mse for ridge on scaled data:	0.01782525371566323

scaling the data isn't necessary. However, when we introduce polynomial terms, we notice that some values in xunscaled become extremely small. As an example, if we take $x = 0.15$ and want to find some term x^6 that becomes roughly 0.00001. This means that the columns of xunscaled have vastly different orders of magnitude, and this calls for scaling to bring them back to a similar scale. If we don't scale, the coefficients ($\boldsymbol{\beta}$) can range widely, from ± 50 in this example. But when we scale the data, this range narrows down to ± 10. Now, this alone might not seem like a strong reason to scale the data, especially when we use plain linear regression (OLS), as it doesn't change the Mean Squared Error (MSE) much. However, things change when we use Ridge regression. In Ridge, the regularization cost depends on the magnitude of β_i. With unscaled data, where β_i varies a lot, some coefficients get penalized more than others. This results in a significantly higher MSE for unscaled data compared to scaled data in Ridge regression. So, to make it easier to tune the regularization parameter (λ) and to ensure fair treatment of coefficients in Ridge regression, it's a good idea to scale the data. This not only simplifies the regularization process but also leads to better predictive performance.

Notice how the first column in Table 2 becomes zero when we scale. This fact is what led us to exclude the first term of the polynomial.

4. Results and Discussion

For the relatively simple Franke's function, we must say that we are quite pleased with the results. We have managed to fit a polynomial of degree 5 to the data with an R^2 score of 0.879 and a MSE of 0.00855 using OLS. Ridge yielded almost identical results when using a small value of $\lambda = 10^{-6}$. Lasso was lagging a bit behind with the best R^2 score of 0.852 and a MSE of 0.0105, also with a small value of $\lambda = 10^{-6}$. Table 4 shows the best scores we managed to get from our models. Table 5 verifies that our implementation of OLS is correct.

Table 4: Best scores from synthetic data

	OLS	Ridge	Lasso
MSE	0.00855	0.00854	0.0105
R^2	0.879	0.869	0.852
λ	N/A	10^{-6}	10^{-6}
Degree	5	5	5

Table 5: Ours Vs SKlearn OLS

SKlearn MSE:	0.008551406119720212
our MSE:	0.00855140611834528
SKlearn R2:	0.8796310495615578
our R2:	0.8796310495803168

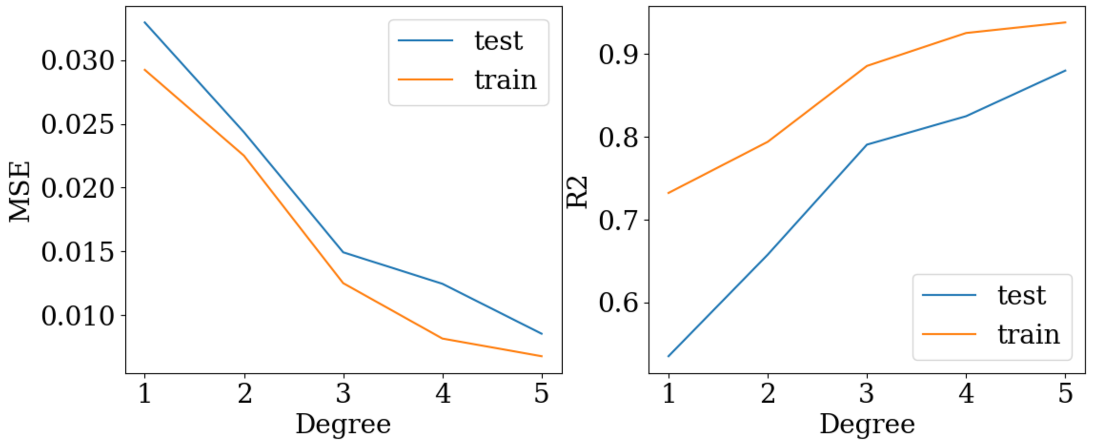

Figure 5: R^2 and MSE with increasing complexity

From Figure 5 we can see that the MSE and R^2 both improve as we increase the model complexity, we have yet to see any signs of overfitting which indicates that we can increase the complexity even further.

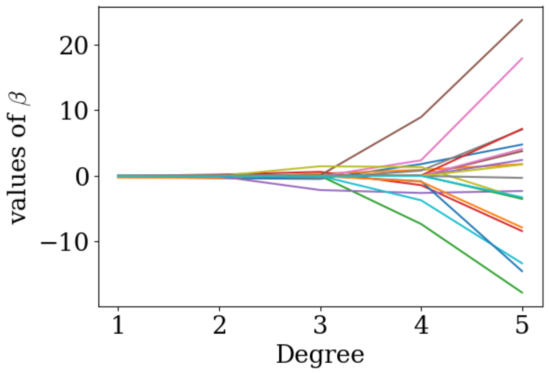

Figure 6: OLS

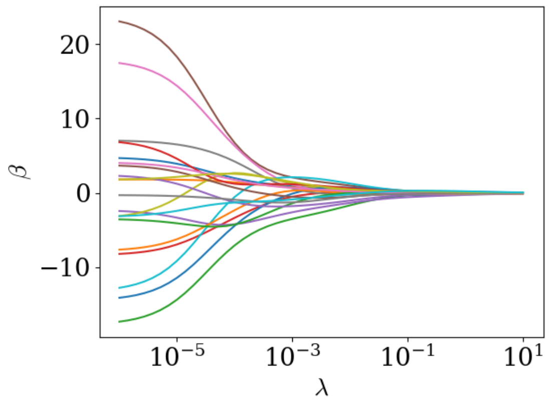

Figure 7: Ridge

We also found that the parameters $\boldsymbol{\beta}$ gets larger as we increase the complexity see Figure 6. On the other hand the parameter λ shrinks the parameters $\boldsymbol{\beta}$ as it increases see Figure 7. This makes perfect sense if we consider the cost function for Ridge regression (Large β_j's are penalized more). The figure makes it clear why $\lambda|\beta|$ is called the shrinkage term.

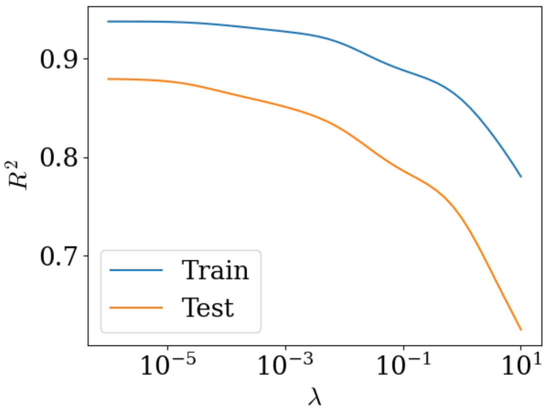

Figure 8: Ridge degree 5

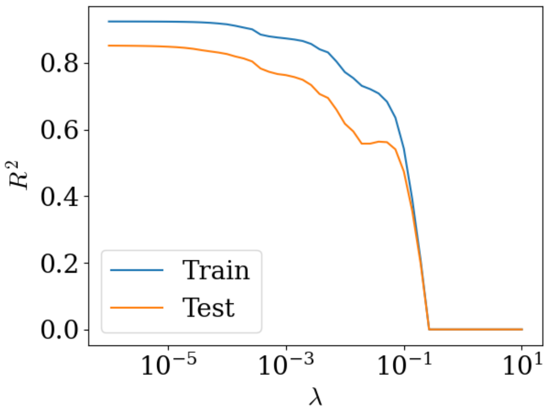

Figure 9: Lasso degree 5

In Figure 5 we noticed that our model had some headroom for increased complexity. In other words not overfitting. As we can see from Figure 8 and Figure 9 increasing λ reduces the R^2 score. This is because we are penalizing the parameters $\boldsymbol{\beta}$ and thus decrease complexity . We don't need to decrease complexity as we show no signs of overfitting. This is probably why we get worse results as we increase λ.

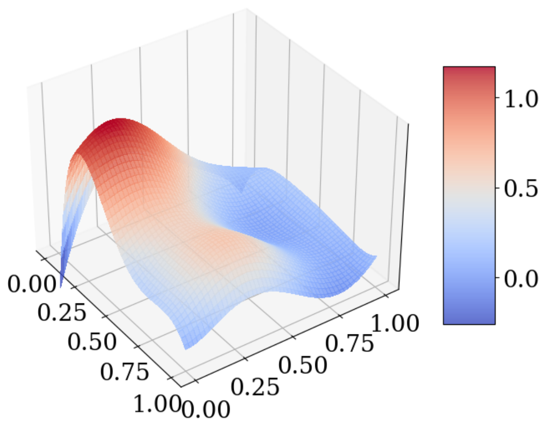

Figure 10: Prediction OLS

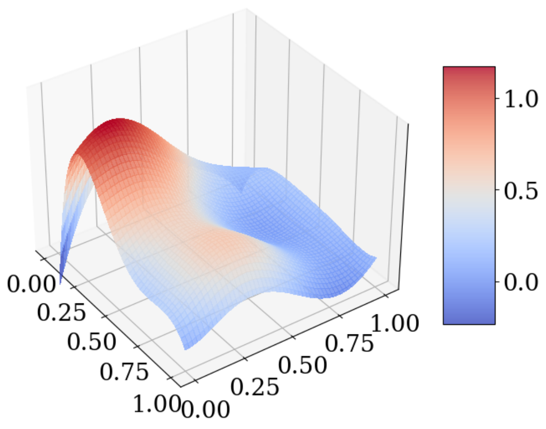

Figure 11: Prediction Ridge

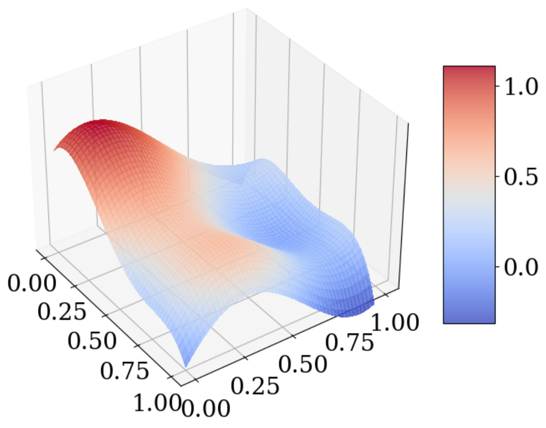

Figure 12: Prediction Lasso

Figure 10, Figure 11 and Figure 12 shows the predictions of our models. All three manage to mimic the true function quite well. Again we can see that OLS and Ridge looks almost identical. Lasso is lagging a bit behind, but still manages to capture the general shape of the true function.

By introducing cross validation and bootstrap we can get a better idea of how our models perform on unseen data. Our choise of k for cross validation is $k = 7$, [2] and we use 100 bootstrap iterations. We will only use bootstrap to showcase the bias-variance trade-off in Section 5. We will use cross validation to find the best parameters for our models.

2. the github repository contains plots and code for finding appropriate k

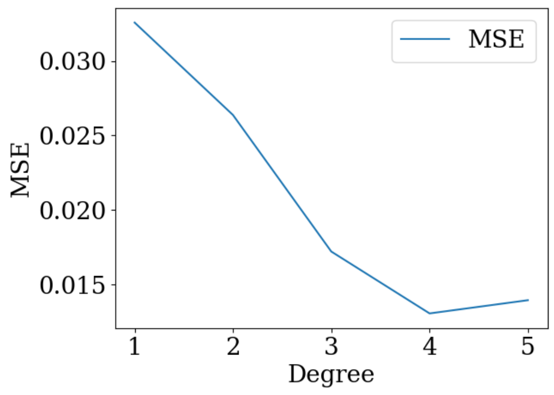

Figure 13: R^2 and MSE with increasing complexity

As we introduce cross validation, we get a slightly different result as we did without. We previously thought that overfitting was something we did't achieve with a polynomial of degree 5. However our eyes have been opened. We can see that the MSE and R^2 score starts to diverge at around degree 5 see Figure 13. For ridge we can see that we maximize the R^2 score at around $\lambda = 10^{-5}$ for polynomial of degree 4 see Figure 14.

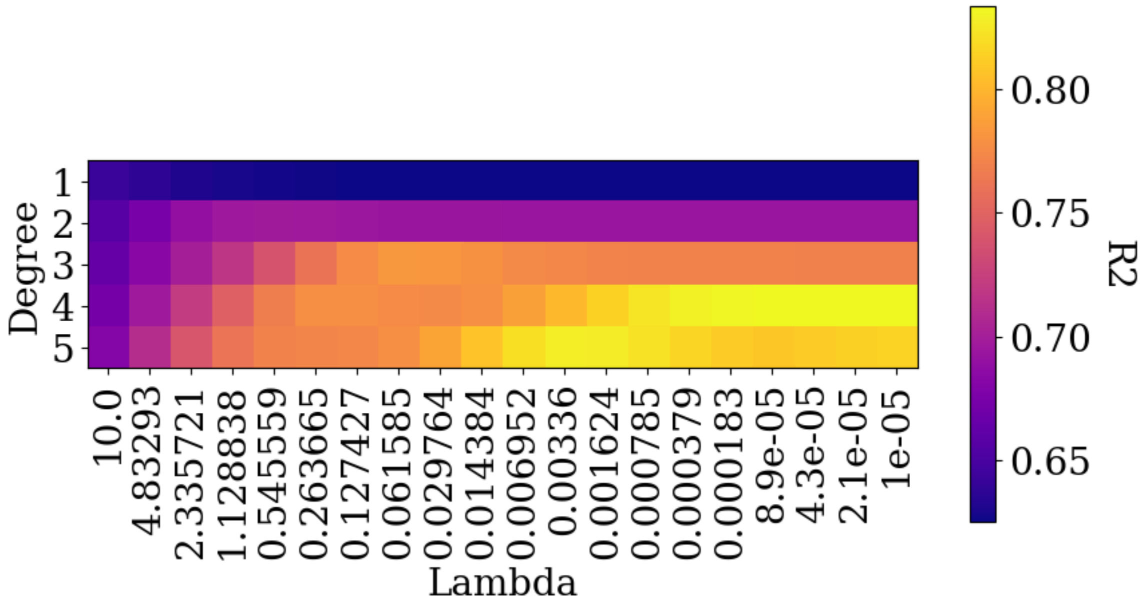

Figure 14: Heatmap Ridge

If we compare our cross-validation to SKLearn's cross-validation we can see similar results, our method seems to lag behind a tiny bit. It is also worth noting that the parameter λ has varying effects on the score

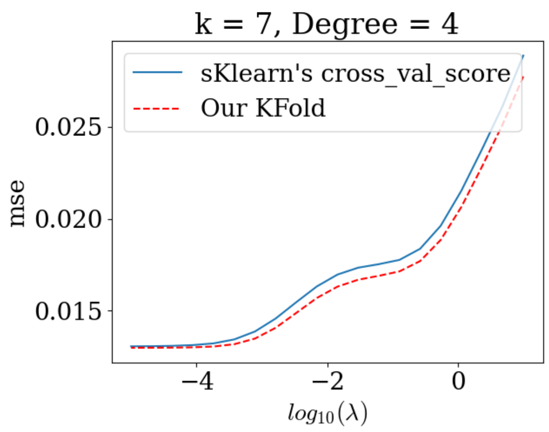

Table 6: Best scores from synthetic data using cross-validation

	OLS	Ridge	Lasso
MSE	0.01305594	0.01306	0.01447
R^2	0.8332	0.0.8331	0.8250
λ	N/A	10^{-5}	10^{-5}
Degree	4	4	5

We end up with the smallest λ for our testing range both with and without cross validation. $\lambda = 10^5$ is fairly close to zero, leading us to suspect that simple OLS is the best model for this data. The scores also reflects this.

In our final experiment we will use the real data. We apply everything we have learned from the synthetic data to the real data. We suspect that a higher degree polynomial will be needed for this. We use polynomials as high as degree 46, and we still don't see any

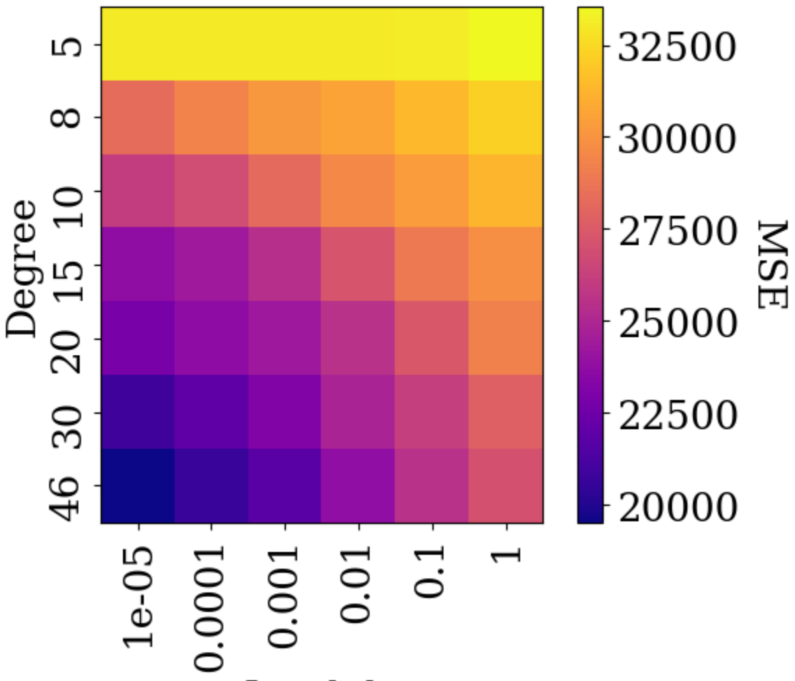

Figure 15: MSE with increasing complexity and λ

signs of overfitting. This is probably because the real data is much more complex than the synthetic data.

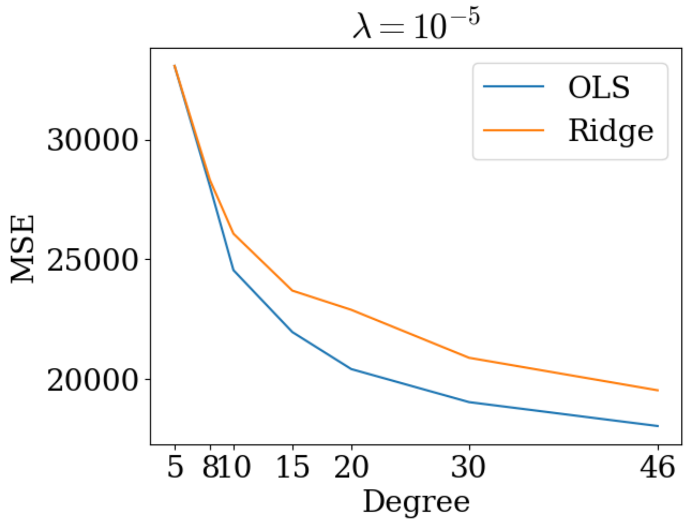

Figure 16: Ols Vs Ridge

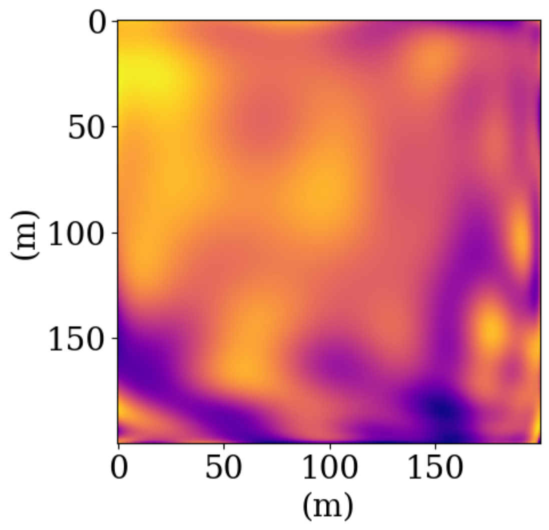

Figure 17: Prediction of the real data top down

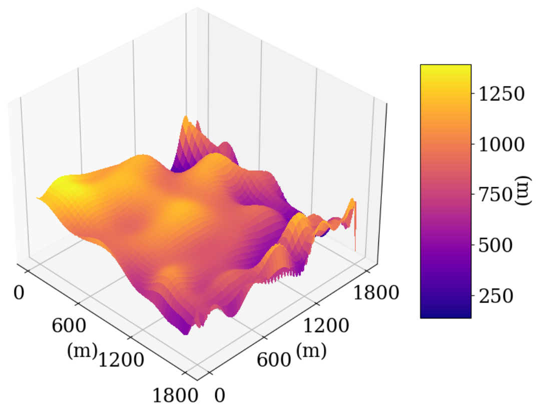

Figure 18: Prediction of the real data

Due to computational limitations lasso did not seem to converge. We did however manage to get a good fit with OLS and Ridge.

Table 7: Best scores from real data

	OLS	Ridge	Lasso
MSE	17453	18692	N/A
R^2	0.7329	0.7292	N/A
λ	N/A	10^{-5}	N/A
Degree	46	46	N/A

5. Bias & Variance

It is possible to perform a so called bias-variance decomposition of our error, see for example Hastie et al. (2009), chapter 7.3 [2]. By doing so we can show that the mean squared error can be written as the following

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

$$\begin{aligned}\mathbb{E}[\mathbf{y}\tilde{\mathbf{y}}] &= \mathbb{E}[\mathbf{f}\tilde{\mathbf{y}} + \epsilon\tilde{\mathbf{y}}] \\ &= \mathbb{E}[\mathbf{f}\tilde{\mathbf{y}}] + \mathbb{E}[\epsilon\tilde{\mathbf{y}}] \\ &= \mathbb{E}[\mathbf{f}\tilde{\mathbf{y}}] + \mathbb{E}[\epsilon]\mathbb{E}[\tilde{\mathbf{y}}] \\ &= \mathbf{f}\mathbb{E}[\tilde{\mathbf{y}}]\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\mathbf{y}^2] &= \mathbb{E}[\mathbf{f} + \epsilon]^2 = \mathbb{E}[\mathbf{f}^2 + 2\mathbf{f}\epsilon + \epsilon^2] \\ &= \mathbb{E}[\mathbf{f}^2] + 2\mathbb{E}[\mathbf{f}\epsilon] + \mathbb{E}[\epsilon^2] \\ &= \mathbb{E}[\mathbf{f}^2] + 2\mathbb{E}[\mathbf{f}]\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2] \\ &= \mathbf{f}^2 + \sigma^2\end{aligned}$$

Bringing it all together we get

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

We need a distribution of $\tilde{\mathbf{y}}$ in order to make it possible to calculate the bias and variance of it. We employ the bootstrap method, which generates multiple samples for $\tilde{\mathbf{y}}$ by using slightly different training data for each sample. In statistical terms, *variance* characterizes the spread or dispersion of our observations. In simpler terms, a model with high variance would exhibit significant variation in the predicted values $\tilde{\mathbf{y}}$ from one test set to another. If we consider

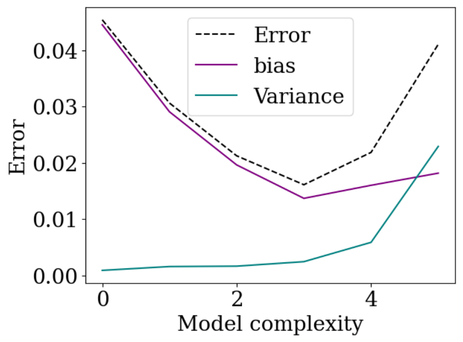

Figure 19:

Figure 20 the blue band depicts our variance, a wider band means more variance. The bias term measures the difference between the actual values $\mathbf{y}$ and the expected values $\tilde{\mathbf{y}}$. This can be seen from the mathematical derivation above. It could also easily be seen from Figure 20 where the black line is the expected value of $\tilde{\mathbf{y}}$ and the red line is $\mathbf{y}$, notice how the closer the black line is to the red line the lower the bias is. These two terms, bias and variance, are somewhat opposing forces. A model with high bias is likely to have low variance, and vice versa. As model designers, we have the flexibility to tune this trade-off. Depending on our priorities, whether we prioritize lower bias or lower variance, we can select an appropriate model to strike the right balance for our specific application. Figure 19 and Figure 20 visualizes the classic bias-variance trade-off.

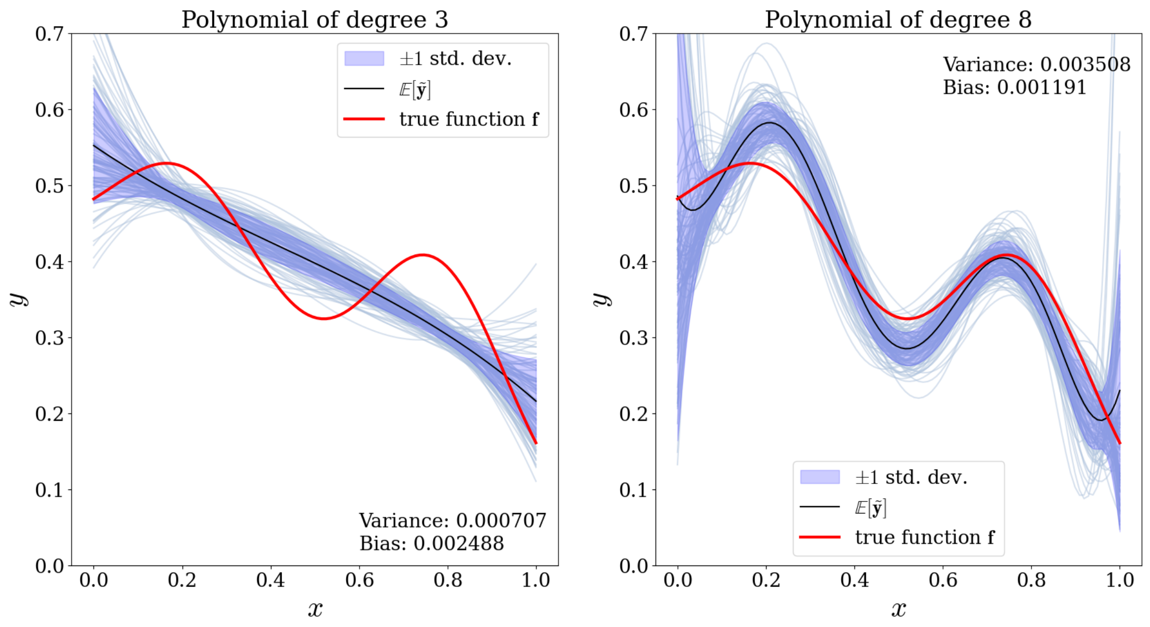

Figure 20: Bias & Variance Showcase, Using 2D Slice Of Our Data

6. Conclusion

Finally we are at the end of our exploration of machine learning and data analysis techniques. We have found that the incorporation of ridge or lasso regularization is dispensable unless encountering issues of overfitting. Nevertheless, the potential for overfitting exists, and the application of ridge or lasso regularization presents a viable remedy for this concern.

In the context of bias and variance analysis, our findings revealed a fundamental opposition between bias and variance. As model designers, we are tasked with making a choice regarding the relative significance of bias and variance based on the specific requirements of our objective. For our case, we found that the bias-variance trade-off was not a significant concern, and we were able to achieve a satisfactory balance between bias and variance without any explicit tuning of the regularization parameter λ. In fact we found that OLS was the best model both cases where tuning of λ is not needed.

Appendix

Finding The Optimal Parameters $\hat{\boldsymbol{\beta}}$

In Section 3.1 we ended up with our optimal parmeters $\boldsymbol{\beta}$ using the MSE as our cost function, in this section we will go into detail about our derivations. The parameters $\boldsymbol{\beta}$ are found by minimizing our cost function (the MSE), this is done by taking the derivative of the MSE with respect to the coefficients β_j and setting it equal to zero.

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=0}^{n-1} \left(y_i - \tilde{y}_i\right)^2$$

can also be written as

$$\begin{aligned} &\min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 \\ &= \frac{2}{n} \mathbf{X}^T \left(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\right) = 0 \\ &= \frac{2}{n} \left(\mathbf{X}^T \mathbf{X}\boldsymbol{\beta} - \mathbf{X}^T \mathbf{y}\right) = 0 \\ \boldsymbol{\beta} &= \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$

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

For the ridge case we have the following cost function

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

We know that

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = \frac{2}{n} \mathbf{X}^T \mathbf{X}\boldsymbol{\beta} - \frac{2}{n} \mathbf{X}^T \mathbf{y}$$

We can also clearly see that

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \lambda ||\beta||_2^2 = \frac{2}{n} \lambda \beta$$

Adding these two we get

$$\begin{aligned} &\frac{\partial \left[\frac{1}{n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2\right]}{\partial \beta} = 0 \\ &= \frac{2}{n} \left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}\right) \boldsymbol{\beta} - \frac{2}{n} \mathbf{X}^T \mathbf{y} \implies \boldsymbol{\beta} = \left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$

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

Confidence Interval

We can make an assumption that there exists a continuous function $f(x)$ and a normal distributed error $\epsilon \sim n(0, \sigma^2)$ which describes our data. [3]

$$y_i = f(x_i) + \epsilon_i$$

We then approximate this function $f(x)$ with our model $\tilde{y}$ from the solution of the linear regression equations, that is our function f is approximated by $\tilde{y}$ where we minimized $(y - \tilde{y})^2$, with $\tilde{y} = \mathbf{X}\boldsymbol{\beta}$ We previously derived the optimal parameters

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

from eq. 1 we can derive the variance of the parameters $\boldsymbol{\beta}$.

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

For ridge regression we get

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

We also have that

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$$

We can use this when we define a so-called confidence interval for the parameters $\hat{\boldsymbol{\beta}}$. the variance of a given parameter β_i is given by the diagonal matrix element of the matrix $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ or $\sigma^2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}$.

$$z_i = \frac{\hat{\beta}_i}{\sqrt{\mathrm{Var}(\hat{\beta}_i)}}$$

(Hastie et al., 2009, p. 48) [2]. A z-score of ± 1.96 corresponds to a confidence interval of 95%

Finding the expectation value and variance of $\hat{\boldsymbol{\beta}}$

We have assumed that y follows some function f with some noise ϵ

$$\mathbf{y} = f(\mathbf{x}) + \epsilon$$

$$\mathbb{E}[\mathbf{y}] = \mathbb{E}[f(\mathbf{x}) + \epsilon] = \mathbb{E}[f(\mathbf{x})] + \mathbb{E}[\epsilon_i]$$

The expected value of ϵ_i is 0, $f(x)$ is a non-stochastic variable and is approimated by $\boldsymbol{X\beta}$. We note the expected value of a single element y_i as

$$\mathbb{E}[y_i] = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\,\boldsymbol{\beta}$$

$$\mathbb{E}[y_i] = \boldsymbol{X_{i,*}\beta}$$

3. This is not an assumption but rather the perfect description of our toy data. When we venture into real world data, we don't know if this is the case. It then becomes an assumption.

The variance of y_i is defined as

$$Var(y_i) = \mathbb{E}\big[(y_i - \mathbb{E}[y_i])^2\big] = \mathbb{E}\left[y_i^2 - 2y_i\mathbb{E}[y_i] + \mathbb{E}[y_i]^2\right] = \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2$$
$$Var(y_i) = \mathbb{E}\big[(\mathbf{X}_{i,*}\beta)^2 + 2\epsilon\mathbf{X}_{i,*}\beta + \epsilon^2\big] - (\mathbf{X}_{i,*}\beta)^2$$
$$Var(y_i) = (\mathbf{X}_{i,*}\beta)^2 + 2\mathbb{E}[\epsilon]\mathbf{X}_{i,*}\beta + \mathbb{E}[\epsilon^2] - (\mathbf{X}_{i,*}\beta)^2$$
$$Var(y_i) = \mathbb{E}[\epsilon^2] = \sigma^2$$

for the expected value of $\boldsymbol{\beta}$ we can insert the definition of $\boldsymbol{\beta}$ from eq. 1

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{Y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta = \beta$$

$$\begin{aligned}
Var(\hat{\beta}) &= \mathbb{E}\{[\beta - \mathbb{E}(\beta)][\beta - \mathbb{E}(\beta)]^T\} \\
&= \mathbb{E}\{[(\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\mathbf{y} - \beta]\,[(\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\mathbf{y} - \beta]^T\} \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbb{E}\{\mathbf{y}\,\mathbf{y}^T\}\,\mathbf{X}\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,\{\mathbf{X}\,\beta\,\beta^T\,\mathbf{X}^T + \sigma^2\}\,\mathbf{X}\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \\
&= \beta\,\beta^T + \sigma^2\,(\mathbf{X}^T\mathbf{X})^{-1} - \beta\,\beta^T \quad = \quad \sigma^2\,(\mathbf{X}^T\mathbf{X})^{-1},
\end{aligned}$$

For ridge we apply the same method

$$\mathbb{E}[\hat{\beta}^{\text{Ridge}}] = \mathbb{E}\left[\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{pp}\right)^{-1}\mathbf{X}^T\mathbf{y}\right] = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{pp}\right)^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{Y}] = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{pp}\right)^{-1}\left(\mathbf{X}^T\mathbf{X}\right)\beta$$

$$\text{Var}(\hat{\beta}^{\text{Ridge}}) = \mathbb{E}\{[\beta - \mathbb{E}(\beta)][\beta - \mathbb{E}(\beta)]^T\}$$
$$\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{pp}\right)^{-1} := \mathbf{A}$$

$$\begin{aligned}
&= \mathbb{E}\left\{\left[\mathbf{A}\mathbf{X}^T\mathbf{y} - \mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)\beta\right]\,\left[\mathbf{A}\mathbf{X}^T\mathbf{y} - \mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)\beta\right]^T\right\} \\
&= \mathbb{E}\left\{\mathbf{A}\mathbf{X}^T\,\mathbf{y}\,\mathbf{y}^T\,\mathbf{X}\,\mathbf{A} - \mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)\beta\,\beta^T\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{A}\right\} \\
&= \mathbf{A}\mathbf{X}^T\,\mathbb{E}\{\mathbf{y}\,\mathbf{y}^T\}\,\mathbf{X}\,\mathbf{A} - \mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)\beta\,\beta^T\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{A} \\
&= \mathbf{A}\mathbf{X}^T\,\left(\mathbf{X}\,\beta\,\beta^T\,\mathbf{X}^T + \sigma^2\mathbf{I}\right)\,\mathbf{X}\,\mathbf{A} - \mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)\beta\,\beta^T\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{A} \\
&= \boldsymbol{A}\boldsymbol{X}^T\,\left(\mathbf{X}\,\boldsymbol{\beta}\,\boldsymbol{\beta}^T\,\mathbf{X}^T + \sigma^2\mathbf{I}\right)\,\mathbf{X}\,\boldsymbol{A} - \boldsymbol{A}\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}\,\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{A} \\
&= \boldsymbol{A}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A} + \sigma^2\boldsymbol{A}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A} - \boldsymbol{A}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A} \\
&= \sigma^2\boldsymbol{A}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A} \\
&= \ \sigma^2[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^T\mathbf{X}\{[\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}]^{-1}\}^T
\end{aligned}$$

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[3] Wessel N. van Wieringen. Lecture notes on ridge regression, 2023.

[4] Brage Wiseth, Felix Cameren Heyerdahl, and Eirik Bjørnson Jahr. MachineLearningProjects. `https://github.com/bragewiseth/MachineLearningProjects`, September 2023.