FYS-STK3155 Project 1

Severin Schirmer

ABSTRACT

The main aim of this project was to study various regression methods, including the Ordinary Least Squares, Ridge, and Lasso regression and to get familiar with model assessment using resampling techniques such as bootstrap and cross validation. We used Franke's function to produce the target data to test our algorithms and perform the model assessments. Finally, we use these techniques to study real world data of terrain elevation outside Stavanger, Norway. For both data sets Ridge regression performed best.

Introduction

Linear regression models were developed in the precomputer age of statistics, but are still relevant and provide a nice entry point into more advanced data analysis and machine learning methods. They are relatively simple and often provide easily interpretable of how the inputs affect the outputs and analytical expressions where other methods are often numerical[1]. The regression models and resampling techniques used in this report can be found in various programming libraries and easily be implemented without much insight into the ideas and mathematics that make them work. Therefore, we wrote our own code for OLS and Ridge with the aim to achieve a deeper understanding of how the models work. However, we use Sci-Kit Learn to implement Lasso as it cannot be solved analytically.

Linear regression models assume a linear relationship between the input values and the output values on the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{1}$$

By using a design matrix, X, for the input data we can use the linear models to perform regression on non-linear relationships.

The report analyses two data sets. A synthetic one sampled from Franke's function, $\mathbb{R}^2 \to \mathbb{R}$, with added noise given by $f(x,y) + \mathcal{N}(0,\sigma^2)$ defined for $x, y \in [0,1]$. The function has been widely used in testing interpolation and fitting algorithms and we used it here to get familiar with how the three linear-regression methods we used work. The second data set is elevation data from an area outside of Stavanger, Norway.

The report starts with an overview of the mathematics behind linear regression and explains the different methods used in the project. The model assessment used is explained before an overview of the implementation is given. Finally, the results for the synthetic and real word data are provided along with the discussion and conclusion.

Method

The problem at hand is modelling a $\mathbb{R}^2 \to \mathbb{R}$ function that can be interpreted as a terrain where the inputs are the coordinates and the output is the elevation. We are using linear regression to solve this problem.

Linear Regression

Linear regression models take in a set of training data, (x, y) and find parameters β that assume a linear relationship between the input and the output. The three methods used in this report, OLS, Ridge, and Lasso, all use the method least squares to find β where the goal is to minimize the residual sum of squares (RSS)[1]

$$RSS = \sum_{i=i}^{N}(y_i - \beta_0 - \sum_{j=i}^{p} x_{ij}\beta_j)^2 \tag{2}$$

The method in simple terms in two dimensions finds the parameters β_0, β_1 which define the line through the data set which minimizes the square of distances from the training data points to the line.

By introducing X, the design matrix, it allows for fitting to non-linear functions such as Franke's function. X is given by a change of the exponent of the input data. A polynomial of degree 3 is then defined as the design matrix, X, and input data (x_1, x_2) on the form

$$X = [x_1 \quad x_2 \quad x_1x_2 \quad x_1^2 \quad x_2^2 \quad x1^2x_2 \quad \ldots \quad x_1^3 \quad x_2^3]$$

where each element is a column vector. X has 10 features denoted by p as seen in in the equation for the RSS (2) (beware that throughout the rest of the report p refers to polynomial order). We can then write the RSS on a more compact form and as a function of β

$$RSS(\beta) = (y - X\beta)^T(y - X\beta) \tag{3}$$

As the goal is to minimize the RSS we find the minimum of $RSS(\beta)$ by, as Hastie et al shows, differentiating with respect to β where the second derivative is positive.[1] By setting the first derivative equal to 0 and solving for β we find the unique solution

$$\hat{\beta} = \left(X^TX\right)^{-1}\left(X^Ty\right) \tag{4}$$

The equation above defines OLS regression and the prediction of the model with parameters $\hat{\beta}$ is given by $\tilde{y} = X\hat{\beta}$.

Shrinkage methods

With many correlated variables in a linear regression model β can be poorly determined and exhibit high variance when training on different subsets of the data.[1] One unreasonably large coefficient can be cancelled out by another large but negative coefficient. Ridge and Lasso (least absolute shrinkage and selection operator) extend on the OLS model by adding a shrinkage parameter λ on the coefficients β to reduce the variance in the model. λ penalizes high values of β and therefore shrinkage methods will produce more stable models by reducing the variance of β.

Ridge regression is given by[1]

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

and β is found with

$$\hat{\beta}^{Ridge} = \left(X^T X + \lambda I\right)^{-1} X^T y \tag{6}$$

We see from equation 5 that the error increases based on λ, however the goal is to produce a more stable model compared to OLS. Where the error will be more consistent across seen and unseen data.[2]

Lasso regression works in a similar way however the shrinkage is scaled differently. Due to the squaring of β in the shrinkage part of equation 5 big values are penalized more than small values. Lasso's paramaters are given by[2]

$$\hat{\beta}^{Lasso} = \underset{\beta}{\operatorname{argmin}} \left(y - X\beta)\right)^T \left(y - X\beta\right) + \lambda|\beta| \tag{7}$$

We see that Lasso shrinks small values more than Ridge. While big values are shrunk relatively less.

Model assessment

Model assessment is key to selecting the optimal model as we use these methods to compare models with different hyper parameters thus finding the optimal one. The methods here are used to avoid models that are overfit and to find the ones that are stable while having a low error. Hyper parameters are the parameters we decide ourselves to control the learning process., in this project these are the polynomial order and the shrinkage parameter λ.

The mean squared error is used as a measure of the quality of a model. It finds the distance from each prediction made by the model to the corresponding point in the data set, squares the distance and then find the mean of all the square distances. It is therefore just the mean of the RSS

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y - \tilde{y})^2 \tag{8}$$

Therefore a model with a high MSE is a poor fit of the data.

Fitting and Overfitting

Before training the model the data set is divided into a training and testing set, usually about 80% for training and 20% for testing. Splitting the data is done to account for the noise that we assume there is in the data set either because the measurements have errors or because the data does not perfectly follow a nice polynomial as is probably the case for the terrain data.

However, if the data did perfectly fit a polynomial of a certain order and the model that was trained on the set was of the same order then there would be no difference between the MSE of the testing and training set. Because there is noise in most data sets we split the data to avoid overfitting, the case where the model fits the training set too well and therefore does not fit the test set. High order polynomials can very accurately fit the data they are trained on as they can perfectly draw a line through all the points if $N \leq p+1$ (N = number of data points, p = polynomial order). Such a model has found a function to fit the noise while the goal is to find the underlying function without the noise. It will exhibit high variance between training sets as the noise varies and perform badly on testing sets. A model is said to be overfit when the MSE of the training and testing set diverge and the training MSE continues to improve while the testing MSE gets worse.

When comparing the models we plot a hyper parameter against the MSE where the hyper parameter can be considered a measure of the complexity of the model.

Bias-Variance Tradeoff

The tradeoff builds on a decomposition of the mean squared error into the bias, variance and noise. We can decompose the MSE as follows

$$\begin{aligned}
E\left[(y-\tilde{y})^2\right] =& E\left[(f+\varepsilon-\tilde{y}+E(\tilde{y})-E(\tilde{y}))^2\right] \\
=& E\left[f^2-2fE(\tilde{y})+E(\tilde{y})^2+\varepsilon^2+\tilde{y}^2-2\tilde{y}E(\tilde{y})+E(\tilde{y})^2+2f\varepsilon-2\varepsilon E(\tilde{y})-2\varepsilon\tilde{y}\right. \\
&\left. -2f\tilde{y}+2fE(\tilde{y})+2\tilde{y}E(\tilde{y})-2E(\tilde{y})^2+2\varepsilon E(\tilde{y})\right] \\
=& E\left[(f-E(\tilde{y}))^2\right]+E(\varepsilon)^2+E\left[\left(E(\tilde{y})^2-\tilde{y}\right)^2\right]+2E\left[(f-E(\tilde{y}))\varepsilon\right]+2E\left[(E(\tilde{y})-\tilde{y})\varepsilon\right] \\
&+2E\left[(E(\tilde{y})-\tilde{y})(f-E(\tilde{y}))\right]
\end{aligned}$$

We use that $E[\varepsilon]=0$ as we assume that the noise is normally distributed around 0.

$$\begin{aligned}
&= E\left[(f-E(\tilde{y}))^2\right]+Var(\varepsilon)+E\left[\left(E(\tilde{y})^2-\tilde{y}\right)^2\right] \\
&= E\left[(f-E(\tilde{y}))^2\right]+E\left[\left(E(\tilde{y})^2-\tilde{y}\right)^2\right]+\sigma^2 \\
&= Bias\left[\tilde{y}\right]^2+Var\left[\tilde{y}\right]+Noise
\end{aligned}$$

The first term is the bias and measures the deviation of the expectation value of our model's prediction from the true value.[2] The second term is the variance and measures how the prediction fluctuates due finite data.[2] Because more complex models exhibit higher variance on small data sets it is often advantageous to use a less complex model partly because it is easier to train, but also because it is less sensitive to noise and will produce a more stable model.

Resampling methods

We use two resampling methods in this project, Bootstrapping and K-fold cross-validation. Resampling methods involve drawing samples from a data set and to fit the model to this subset or changed data set. We are then able to perform more advanced analysis of the model including the bias-variance tradeoff discussed previously or find the variance of β another measure of the stability of the model.

In bootstrapping samples are drawn from the data set with replacement to produce a new data set of equal size. Due to the replacement data sets are likely to contain repeats of data points and to not include the entire set. A model is then trained on the data set and a prediction is made on a separate testing set. This is done repeatedly for a decided upon number of times and the predictions made by each model and the model's parameters are saved. We can then find the variance of the model by finding how the prediction in a specific data point varies across the models and take the expectation value of the variance of all predicted values. The bias is more intuitive and is subtracting the mean predicted value in each data point from the target value. For OLS it is also possible to get a analytical expression for the variance of β given by $Var(\beta) = \sigma^2(X^TX)^{-1}$.[3] It is then possible to find a 95% confidence interval for β. For Ridge and Lasso one must do a percentile interval, however, this is not covered in this project. If done, one can compare the accuracy and stability of the models.

K-fold cross-validation is less computationally expensive than bootstrapping. In this project we worked with 5 folds due the small data set. K-fold cross validation splits the data set into K folds and for each iteration leaves one partition out. The model is trained on the remaining parts and tested on the one left out. The MSE is calculated in each iteration and finally one takes the mean of the MSE to produce a score. One assumes that outliers will cancel each other out across the iterations.

Model selection

Model selection is often done in none rigorous fashion where or "by the the eye". However, we can also employ more sturdy techniques to decide on the optimal model based on our model assessment. In this project we employed both approaches. When selecting a model the aim is to find a model with high prediction capability while not being too complex to avoid overfitting.

We based most of our selection on the Kfold MSE where with the models trained on Franke's function we used a mix of lowest MSE and "by the eye". However, the more rigorous approach is to choose the least complex model within one standard deviation of the error of the best performing one.[1] This is employed on the terrain data.

Implementation

The files can be found at my GitHub:

https://github.uio.no/severs/FYS-STK3155/tree/master/Project1

The method was implemented in Python mainly using the library Numpy with Matplotlib and Seaborn for plotting. Sci-Kit Learn is used for Lasso.

The code can be divided into three sections:

1. Setup - Data handling and base functions
2. Model assessment - Assessments of models using bootstrap and cross validation, including model selection.
3. Testing of selected models - plotting predictions and comparison of MSE and R2 scores

The analysis of the terrain is more succinct in its implementation with more generalizable functions.

A notable difference between the data handling for Franke's function and for the terrain is the way the data was split. Franke's function is split using our own split function that works the same way as Sci-Kit learn's function `train_test_split`. However, it is easier to ensure that the split for different models are the same as the seed is set within the function.

For the terrain we opted for a different approach where we selected evenly spaced points throughout the area, shifting the grid for the validation set and shrinking it and shifting again for the test set.

Results

Modelling Franke's Function

The three regression methods were first used to model a noisy data set from Franke's function seen in figure 1.

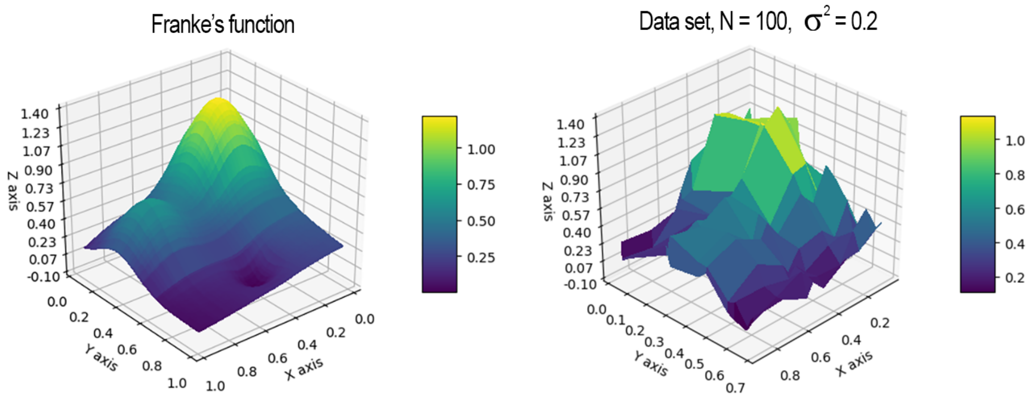


Figure 1. The right graph shows a random subset of points from Franke's function (left) for $x, y \in [0,1]$ with noise $\mathcal{N}(0, 0.2)$.

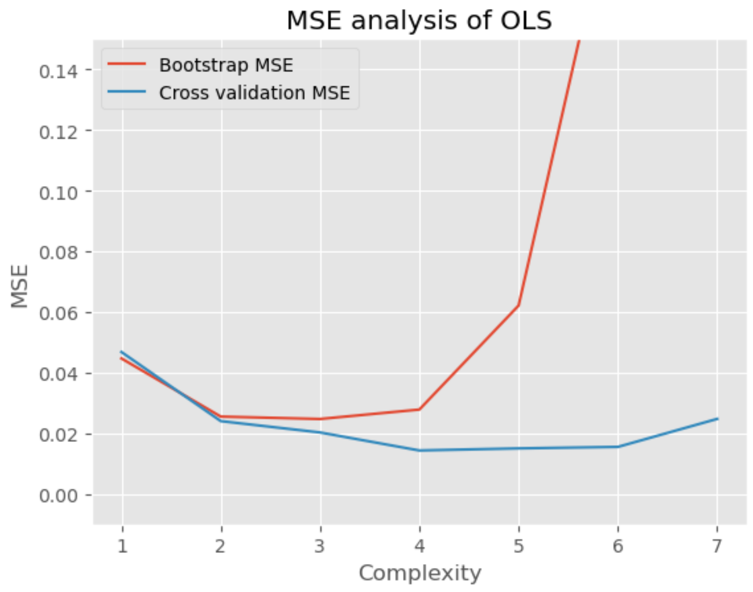


Figure 2. Bootsrap estimate with 1000 iterations vs fivefold cross validation. Bootstrap overfits for lower polynomial orders than cross validation.

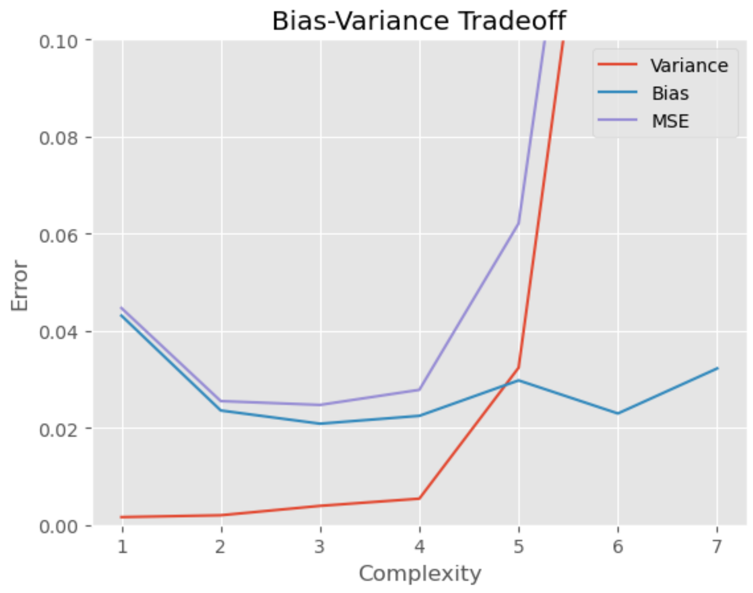
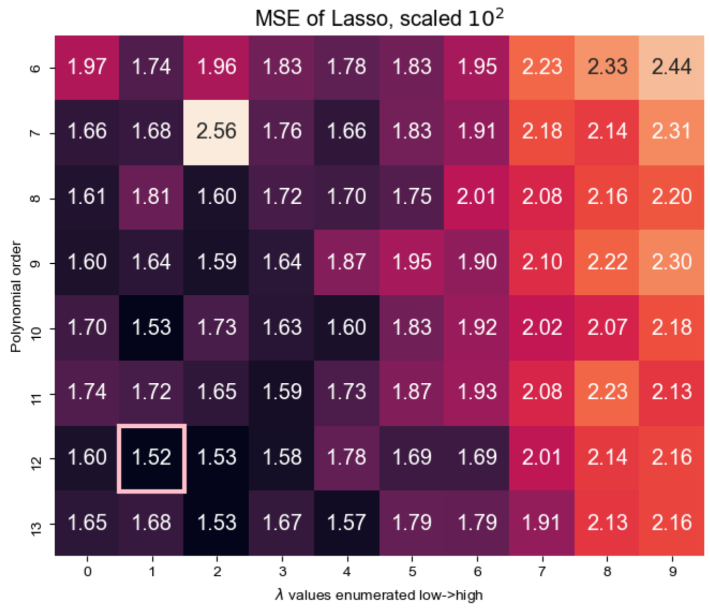


Figure 3. The estimate shows that the model becomes unstable for $p = 5$ and and the minimum MSE is for $p = 3$

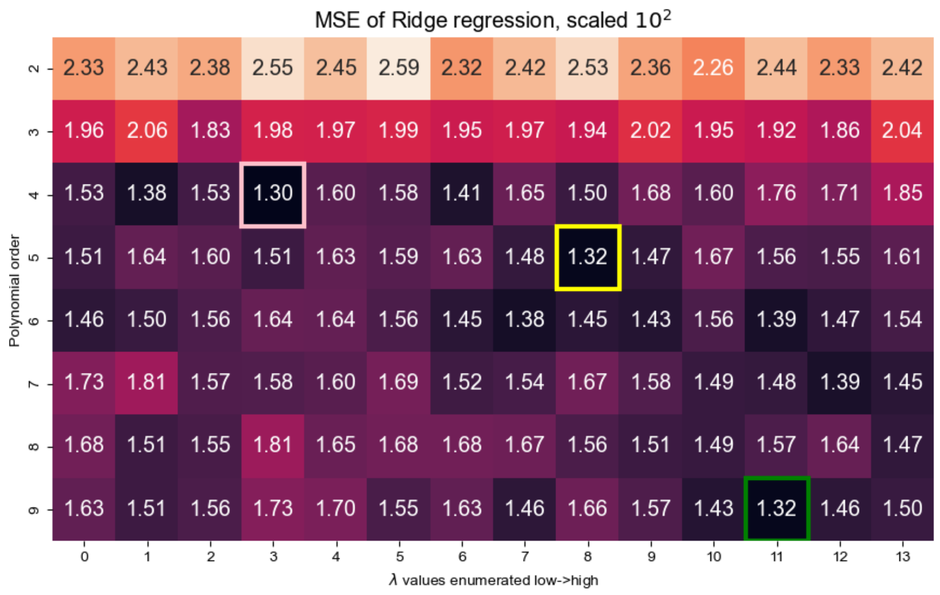
p	MSE	R2
1	0.0468	0.3948
2	0.0240	0.6652
3	0.0202	0.7531
4	**0.0143**	0.7868
5	0.0150	**0.8043**
6	0.0155	0.7843
7	0.0247	0.5998

Table 1. Estimates of MSE and R2 score for OLS polynomials of order p. The optmimum scores are bold and the estimates were obtained with fivefold cross validation.



Polynomial order	0	1	2	3	4	5	6	7	8	9
6	1.97	1.74	1.96	1.83	1.78	1.83	1.95	2.23	2.33	2.44
7	1.66	1.68	2.56	1.76	1.66	1.83	1.91	2.18	2.14	2.31
8	1.61	1.81	1.60	1.72	1.70	1.75	2.01	2.08	2.16	2.20
9	1.60	1.64	1.59	1.64	1.87	1.95	1.90	2.10	2.22	2.30
10	1.70	1.53	1.73	1.63	1.60	1.83	1.92	2.02	2.07	2.18
11	1.74	1.72	1.65	1.59	1.73	1.87	1.93	2.08	2.23	2.13
12	1.60	1.52	1.53	1.58	1.78	1.69	1.69	2.01	2.14	2.16
13	1.65	1.68	1.53	1.67	1.57	1.79	1.79	1.91	2.13	2.16

Figure 5. MSE estimate of Lasso with cross validation. Box: pink = minimum MSE.



Polynomial order	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	2.33	2.43	2.38	2.55	2.45	2.59	2.32	2.42	2.53	2.36	2.26	2.44	2.33	2.42
3	1.96	2.06	1.83	1.98	1.97	1.99	1.95	1.97	1.94	2.02	1.95	1.92	1.86	2.04
4	1.53	1.38	1.53	1.30	1.60	1.58	1.41	1.65	1.50	1.68	1.60	1.76	1.71	1.85
5	1.51	1.64	1.60	1.51	1.63	1.59	1.63	1.48	1.32	1.47	1.67	1.56	1.55	1.61
6	1.46	1.50	1.56	1.64	1.64	1.56	1.45	1.38	1.45	1.43	1.56	1.39	1.47	1.54
7	1.73	1.81	1.57	1.58	1.60	1.69	1.52	1.54	1.67	1.58	1.49	1.48	1.39	1.45
8	1.68	1.51	1.55	1.81	1.65	1.68	1.68	1.67	1.56	1.51	1.49	1.57	1.64	1.47
9	1.63	1.51	1.56	1.73	1.70	1.55	1.63	1.46	1.66	1.57	1.43	1.32	1.46	1.50

Figure 4. MSE estimate of Ridge with cross validation. Boxes: pink = minimum, green = second minimum, yellow = third minimum. Some p-values and λ-values are cropped for legibility.

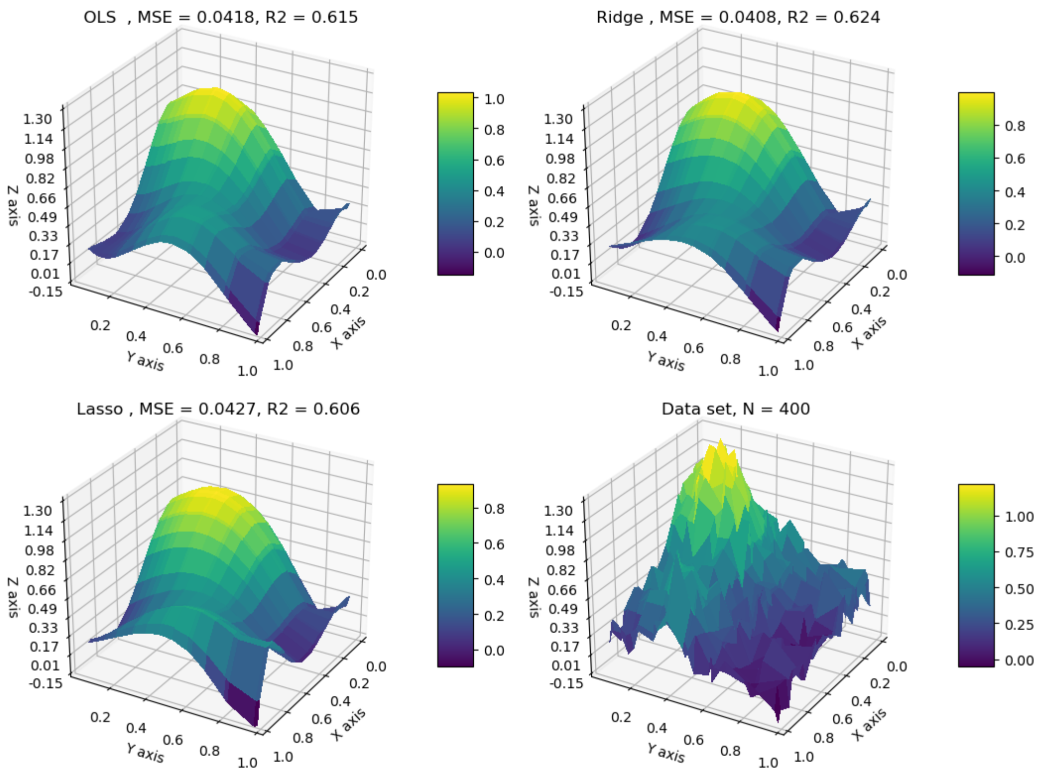


Figure 6. The three models are quite similar in their predictions with Lasso faring slightly better. Looking at Figure 1 we can understand the curved tops of the predictions from the training data as it had a flatter shape than the usual function.

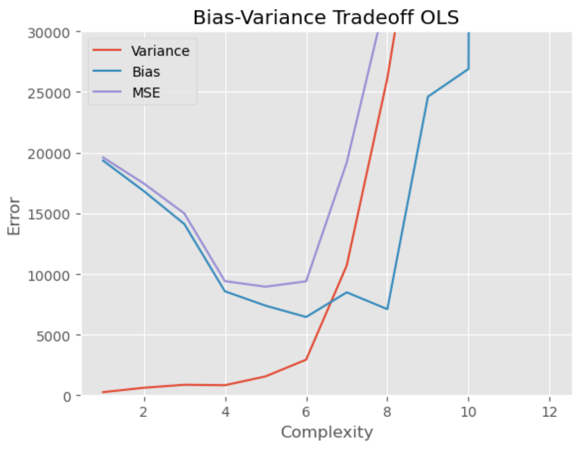


Figure 7. Bootstrap with 100 iterations and minimum at MSE for the polynomial of the fifth order.

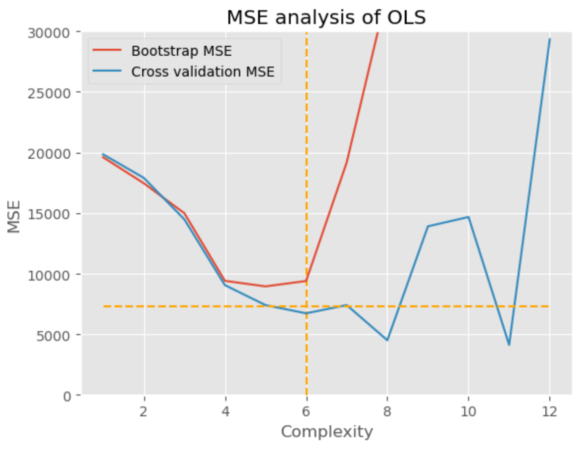


Figure 8. The bootstrap MSE estimate was obtained with 100 iterations and the cross validation MSE estimate was obtained with ten folds. The least complex model within one standard error of the best was chosen, indicated by the orange vertical broken lines.

Model	p	λ
OLS	6	n/a
Ridge	11	10^{-10}
Lasso	7	10^{-5}

Table 2. Selected models are the least complex model within one standard error of the best. p = Polynomial order.

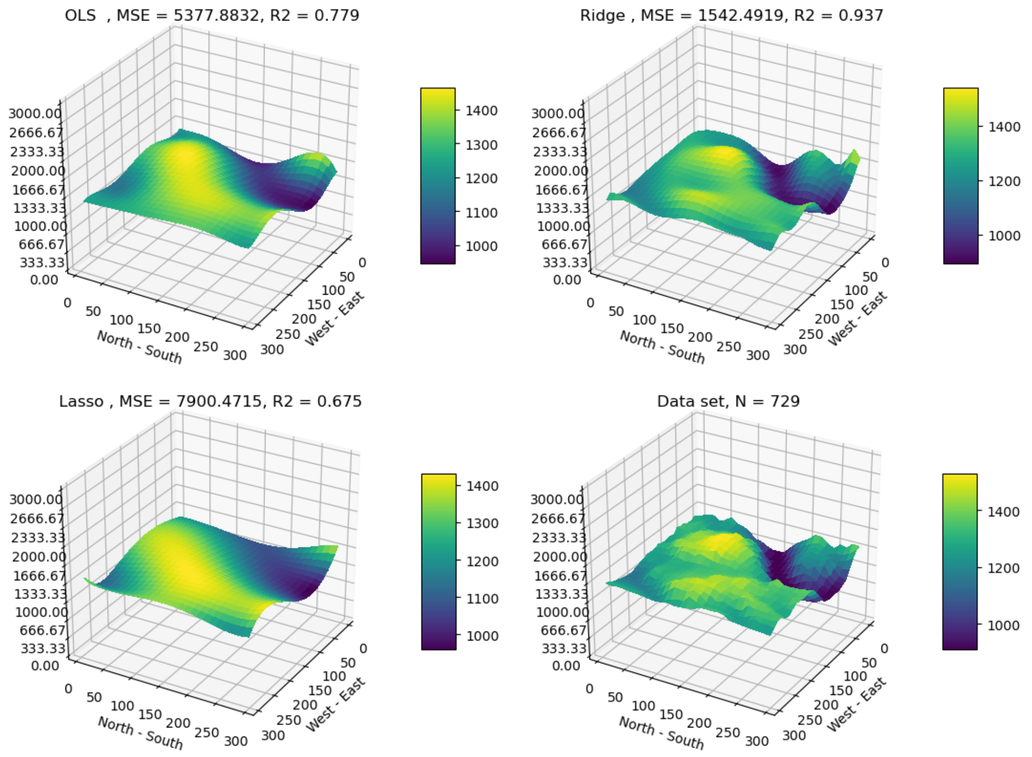


Figure 9. The bootstrap MSE estimate was obtained with 100 iterations and the cross validation MSE estimate was obtained with ten folds. The least complex model within one standard error of the best was chosen, indicated by the orange vertical broken lines.

Discussion

Model selection

Model selection is a tricky sport. Due to the implementation of bootstrap it systematically estimated lower MSEs than cross validation. This is due to less of the data being used in bootstrap as the validation set is set aside for model assessment. For cross validation the data set is almost twice the size because we use both the validation and training set to train the model while one fold is left out in each iteration to test it.

Franke's function

Ordinary Least Squares: $p = 4$

Ordinary least squares seems to make the best predictions for polynomial degrees $(p) \in [3,5]$. The $R2$ score is highest for $p = 5$, however the bias-variance analysis shows that the model is overfit for $p = 5$ and has minimum MSE at $p = 4$. We decide that the polynomial of the 4^{th} order is our selected model because the cross validation reaches its minimum here and its a compromise between the $R2$ score and the bias-variance analysis.

Ridge: $p = 5$, $\lambda = 2.15 \cdot 10^{-3}$

The most obvious candidate is the model with the lowest MSE indicated by pink in figure 4. However the λ value is $2.15 \cdot 10^{-4}$, which means the shrinkage is quite small. Because this model had the same polynomial order as our selected OLS model we looked at the second and third minima as candidates. These two score almost the same, but the second minimmum is a quite complex model and we therefore choose the third which has $p = 5$ and $\lambda = 2.15 \cdot 10^{-3}$ as we assume it will generelize better.

Lasso: $p = 12$, $\lambda = 1.47 \cdot 10^{-4}$

Lasso performed poorly for $p \leq 5$ therefore we tested for higher polynomial orders up to the arbitrary $p = 13$ and found a minimum for $p = 12$, $\lambda = 1.47 \cdot 10^{-4}$.

Testing models

For Franke's function we have unlimited data and here we took the freedom to produce a new data set to test the three models against each other. Therefore, we could also produce enough data points to make comprehensible plots of the predictions. The models were still trained on the training set from the initial data set however.

Terrain data

For the terrain data the selection was automated as described in the method section. Plots from Lasso and Ridge can be seen in the appendix. However, the selection of simplest model was made knowing that the simplest model was in the first row. If it weren't the implementation might have selected a model that was not the simplest or arguably more complex than the best performing one. Model complexity usually ascends from top left to bottom right of figures like Figure 10. However, this depends on the λ values used. A better implementation of the selector would select models in true descending order of complexity. However, this was not needed in this time.

Conclusion

We found that Ridge regression performed best across the two data sets. However, we saw that due to the small size of the data set from Franke's function, the data set missed important features that were evident in the testing set and in most random samples from the function.

Lasso performs worse than the other two models, likely due to the shrinkage working linearly and not allowing the small coefficients to affect the predictions.

Moving forward, one could eliminate the need for as many functions as used in this project. The relatively high number occured due to the difficulty of generalising assessment functions for one and two hyper parameters and implementations of the functions analytically versus in Sci-Kit Learn. The easiest way to solve this is most likely by using classes to implement the functions. There is also a possibility of increasing the efficiency of the design matrix by computing the design matrix for the maximum polynomial degree and then indexing the columns used for lower polynomial degrees.

An interesting assessment that was not done in the project is finding confidence intervals of the Ridge and Lasso models. This is another estimate of the stability of the models and can probably easily be implemented with percentile intervals using bootstrapping.

References

1. Hastie, T., Tibshirani, R. & Friedman, J. *Linear methods for regression*, 43–100 (Springer, 2009), 2 edn.

2. Mehta, P., Wang, C.-H., Day, A. G. R. & Richardson, C. A high-bias, low-variance introduction to machine learning for physicists. *Phys. Reports* **810**, 1–124, DOI: https://doi.org/10.1016/j.physrep.2019.03.001 (2019).

3. Hjorth-Jensen, M. Week 36: Resampling techniques and ordinary least square (2020). Available at https://compphysics.github.io/MachineLearning/doc/pub/week36/html/week36-reveal.html.

Appendix

β	Lower	Upper
0.46	0.44	0.48
2.44	1.68	3.20
0.73	-0.25	1.71
-9.10	-11.80	-6.40
-2.70	-3.97	-1.43
0.27	-3.59	4.13
10.21	6.48	13.93
3.81	2.19	5.42
2.74	1.15	4.32
-3.71	-8.97	1.55
-3.70	-5.49	-1.92
-2.16	-3.01	-1.31
-0.21	-0.83	0.41
-1.52	-2.34	-0.69
2.67	0.32	5.02

Table 3. The 95% confidence intervals for the OLS model trained on Franke's function of polynomial order 4. We see that higher absolute value of β results in wider intervals.

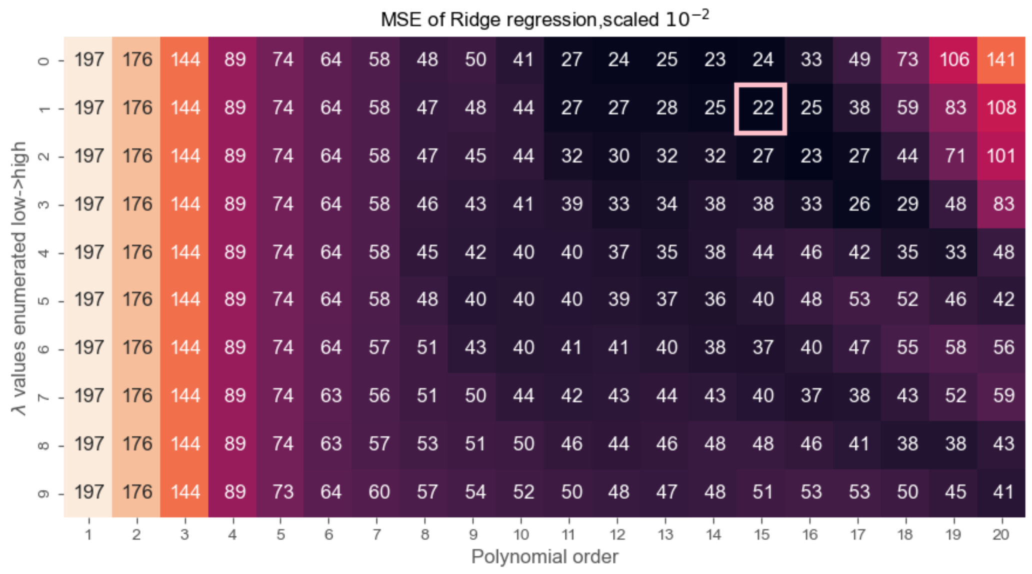


Figure 10. MSE scores for 7 fold cross validation. The minimum MSE of 2220 is indicated by the pink box. All MSEs lower than 2896 are within one standard deviation of the minimum. The model with polynomial order 11 and λ-index 0 ($\lambda = 10^{-10}$) was selected.

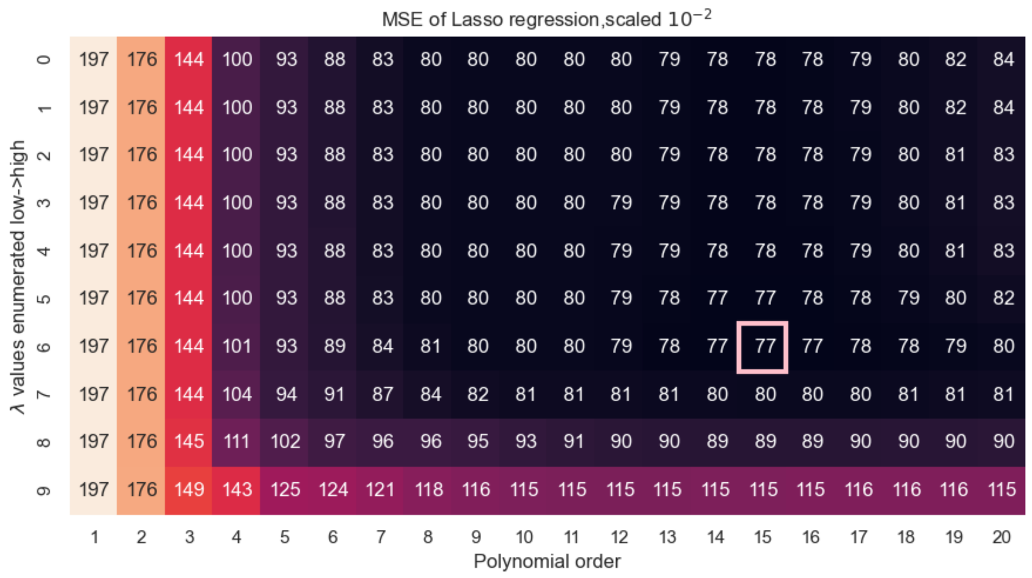


Figure 11. MSE scores for 7 fold cross validation. The minimum MSE of 7709 is indicated by the pink box. All MSEs lower than 8712 are within one standard deviation of the minimum. The model with Polynomial order 7 and λ-index 0 ($\lambda = 10^{-5}$) was selected.