Department of Physics
The Faculty of Mathematics and Natural Sciences
University of Oslo

Regression analysis and resampling methods

FYS-STK4155 Project 1

Fall semester of 2024

Aleksander Berge Gursli - aleksbg

Abstract

In this project report I explored different linear regression methods to get a detailed study into regression analysis. Two different datasets are used, one that is calculated and generated using the Franke function, and one real dataset in the form of digital terrain data from Norway. Firstly on the generated dataset, a standard ordinary least square regression analysis is performed, the evaluation of the mean squared error and R^2 is also included. This is then repeated for both Ridge and Lasso regression before I continue with different resampling techniques (bootstrap and cross-validation). For the Franke function generated data I end up concluding (based on my current results) that all models perform similarly well if the variables are tuned correctly for the given function. Finally, to have an analysis of real data, everything is reapplied on the real digital terrain dataset - where in my results the OLS performed the best.

Contents

1 Introduction

As one of the central methods in machine learning, regression analysis is an important and powerful tool. As probably the first and at the very least one of the most common form of regression analysis, linear regression is important to delve into and understand. Specifically in this report, I evaluate and perform three different types of linear regression methods. Ordinary least squares, ridge and lasso regression. Furthermore, as this is a project task as the first part of a 3-project in a machine learning course - it's an important stepping stone to understanding and implementing more complex methods later on.
The methods mentioned will be implemented on two sets of data; data generated via the Franke function, and real terrain data from a region by Stavanger, Norway. This will be done to see what manner of regression is best suited the different types of data, and the evaluation of the methods will be done by usage of regular statistical metrics you often see in regression - Mean Square Error (MSE) and R^2 score. Furthermore, the effect of the bias-variance trade-off will be studied by implementing bootstrap resampling (for OLS only). For all methods of regression, k-fold cross validation will be implemented as another resampling technique to compare with the results of bootstrapping.
All these metrics and methods will be inspected to determine what the most fitting manner of linear regression is for both the Franke function generated data, and the real terrain data. The report is structured into 4 sections (including introduction), covering in section 2 the methods and theory part of the project - here are the answers to all theory tasks. Section 3 is comprised of the results and brief discussion, while section 4 is the conclusion. Additionally there are 2 appendices - A where I cover the data and python libraries used. And appendix B where the link to my github repo with code and results can be found.

2 Methods and theory tasks

In line with our teaching material, I use $\tilde{\mathbf{y}}$ and not $\hat{\mathbf{y}}$ as notation for the predicted data.

2.1 Linear Regression Analysis

In linear regression, and thus in this report, one aims to investigate the relationship between real or observed data and that of predicted or modelled data. Importantly, one wants to evaluate the choice of model and its ability to further predict data. Given a dataset of observed values $\mathbf{y}$, one would model a dataset of predicted values $\tilde{\mathbf{y}}$, and then evaluate how good of a prediction or model it really is. Even

though there are many different methods for evaluating the predictive ability or performance of the model, mean squared error (MSE) and the coefficient of determination (R^2 score) are the ones featured here.
The mean square error is calculated by:

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

And given:

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

the R^2 score is calculated by:

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

2.1.1 Project part d) Theory task

As part of the project we were tasked with showing a few calculations in regards to variance, expected and mean values. Under the assumption of the existence of a continuous function and a normal distributed error, $f(\mathbf{x})$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ respectively. This in turn describes the real data:

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

To approximate $f(\mathbf{x})$, the model $\tilde{\mathbf{y}}$ from the solution of ordinary least squares (i.e. after minimizing $(\mathbf{y} - \tilde{\mathbf{y}})$) can be used, and then:

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

Where $\mathbf{X}$ is the design matrix of dimensionality $n \times p$, with n being the number of data points and p the predictors. With the response variable $\mathbf{y}$, and the assumption of a linear relationship between $\mathbf{X}$ and $\mathbf{y}$, we have the regression parameters as $\beta = [\beta_{\mathbf{0}}, ..., \beta_{\mathbf{p-1}}]^{\mathbf{T}}$. And for the response variables, we then have:

$$y_i = \beta_0 x_{i0} + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_{n-1} x_{in-1} + \varepsilon_i,$$

which on vector form can be written as:

$$y_i = \mathbf{x_i^T}\beta + \varepsilon_i = (\mathbf{X}\beta)_i + \varepsilon_i$$

And such, to find $\mathbb{E}(y_i)$, we find $\mathbb{E}((\mathbf{X}\beta)_i + \varepsilon_i)$. By definition of the error ε and the expected value of an non-stochastic variable we have:

$$\begin{aligned}\mathbb{E}((\mathbf{X}\beta)_i + \varepsilon_i) &= \mathbb{E}((\mathbf{X}\beta)_i) + \mathbb{E}(\varepsilon_i) \\ &= (\mathbf{X}\beta)_i \\ &= \mathbf{X}_{i,*}\,\beta\end{aligned}$$

Based on this, we move on to find the variance of y_i:

$$\begin{aligned}Var(y_i) &= \mathbb{E}[(y_i - \mathbb{E}(y_i))^2] \\ &= \mathbb{E}(y_i^2) - (\mathbb{E}(y_i))^2\end{aligned}$$

Using the result from $\mathbb{E}(y_i)$:

$$
\begin{aligned}
&= \mathbb{E}((\mathbf{X_{i,*}}\,\beta + \varepsilon_i)^2) - \mathbf{X}_{i,*}\,\beta \\
&= \mathbb{E}((\mathbf{X}_{i,*}\,\beta)^2 + 2\varepsilon_i(\mathbf{X}_{i,*}\,\beta) + \varepsilon_i^2) - (\mathbf{X}_{i,*}\,\beta)^2 \\
&= 2\mathbb{E}(\varepsilon_i)(\mathbf{X}_{i,*}\,\beta) + \mathbb{E}(\varepsilon_i^2) \\
&= \mathbb{E}(\varepsilon_i^2) \\
&= Var(\varepsilon_i) = \sigma^2
\end{aligned}
$$

2.1.2 Project part d) Theory task - OLS

We were further tasked to show that,

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}.$$

Given the OLS expressions for the optimal parameters $\hat{\boldsymbol{\beta}}$:

Since the minimized OLS cost function gives these expressions for $\hat{\boldsymbol{\beta}}$, and is defined by;

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

We can then take the expectation of $\hat{\boldsymbol{\beta}}$, and we get:

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\right) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}),$$

and due to the fact that $\mathbb{E}(\boldsymbol{y}) = \boldsymbol{X}\boldsymbol{\beta}$, we can reduce to;

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{\beta}.$$

And then we use this result to show the next part of the theory assignment: that the variance of $\boldsymbol{\beta}$ is

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

which we can rewrite as:

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left((\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}}))^2\right).$$

And using the result from above: $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ and $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, we can rewrite:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) = \boldsymbol{\beta} + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\varepsilon}.$$

Which means that the variance of $\hat{\boldsymbol{\beta}}$ becomes

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\varepsilon}\right)^2\Big) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

And then, due to $\mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T) = \sigma^2\boldsymbol{I}$, we finally show that

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

All results above is based on (T. Hastie, 2009) and (Hjorth-Jensen, 2021).

2.1.3 Ridge

2.1.4 Lasso

2.2 Scaling

Regarding scaling, as stated in the appendix - due to our implementation of the Franke function being inherently scaled - I don't scale the data for this part of the assignment. On the other hand, for the real terrain data I scale the data to the general standard of scikit-learn's standardscaler, which works by subtracting the mean and dividing by the standard deviation (i.e. scaling to unit variance) - see StandardScaler docs at (Pedregosa et al., 2011).

2.3 Bias-variance trade-off - theory task

We were again tasked with a theory assignment regarding bias-variance trade-off. Again, we are to use the results and rules in part d) together with this part (I've here copied the task assignment directly so that it's clear):
The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the mean squared error via the so-called cost function

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

Here the expected value $\mathbb{E}$ is the sample value.

Show that you can rewrite this in terms of a term which contains the variance of the model itself (the so-called variance term), a term which measures the deviation from the true data and the mean value of the model (the bias term) and finally the variance of the noise. That is, show that

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \text{Bias}[\tilde{y}] + \text{var}[\tilde{y}] + \sigma^2,$$

with

$$\text{Bias}[\tilde{y}] = \mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right],$$

and

$$\text{var}[\tilde{y}] = \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] = \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2.$$

To tackle this, we begin by way of the MSE expression:

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

We start by expanding;

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

Due to $\mathbb{E}\left[\boldsymbol{\epsilon}^2\right] = \sigma^2$, we can rewrite as follows;

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(f(\boldsymbol{x}) - \tilde{\boldsymbol{y}})^2\right] + \sigma^2$$

We can then tackle the first term $\mathbb{E}\left[(f(\boldsymbol{x}) - \tilde{\boldsymbol{y}})^2\right]$ by adding and subtracting $\mathbb{E}[\tilde{\boldsymbol{y}}]$, leaving us with:

$$\mathbb{E}\left[(f(\boldsymbol{x}) - \tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(f(\boldsymbol{x}) - \mathbb{E}[\tilde{\boldsymbol{y}}] + \mathbb{E}[\tilde{\boldsymbol{y}}] - \tilde{\boldsymbol{y}})^2\right]$$

By expanding this new right-hand term, we end up with three terms :

$$\mathbb{E}\left[(f(\boldsymbol{x}) - \tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(f(\boldsymbol{x}) - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\right] + \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\right]$$

And looking up, we can see that due to:

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

We have ended up with the first right-hand term being equal to $\text{Bias}[\tilde{y}]$ and $\text{var}[\tilde{y}]$ and thus going back to the full equation we end up with:

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \text{Bias}[\tilde{y}] + \text{var}[\tilde{y}] + \sigma^2$$

Going on, we were also tasked with explaining the terms and their interpretations: The bias is a measure of how much discrepancy exists between the expected value of the model $\mathbb{E}[\tilde{\boldsymbol{y}}]$ and the true function $f(\boldsymbol{x})$. If the bias is high, it can indicate that the model is not a good fit - due to underfitting.

The variance is a measure of how much discrepancy exists between the predictions $\tilde{\boldsymbol{y}}$ and their expected value, i.e. how they fluctuate around the $\mathbb{E}[\tilde{\boldsymbol{y}}]$. High variance also indicates that the model is not a good fit, and that it is overfitting due to fluctuations in the data.

This bias-variance trade-off then helps us understand the performance of a

model. It can be called a ”dance” of sorts between under- and overfitting. Where a model with high bias is too simplistic - which lead to underfitting of the data, a model with high variance is too complex - leading to overfitting of the the data. This means that the best way to handle the trade-off is to find a model that ensures balance between bias and variance (and thus over- and underfitting), which would lead to minimization of the error as a whole.

As in the previous paper and pencil task, all these answers are based on (T. Hastie, 2009) and (Hjorth-Jensen, 2021).

2.4 Resampling

As stated, we are in this project using two methods of resampling - bootstrapping and K-fold cross validation. Resampling is a good method to take advantage of due to how it works. As in the meaning of the word, resampling grants better evaluations of models, and improves our understanding of their performance. This is done by resampling, or for lack of a better word going over samples from the training set again-and-again.

2.4.1 Bootstrapping

2.4.2 K-fold cross validation

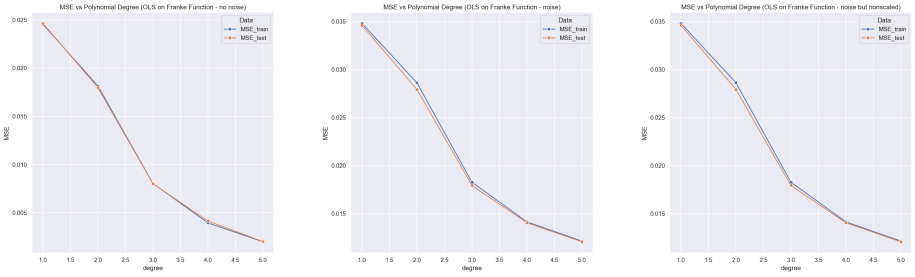
3 Results and discussion

3.1 Franke function

The first part of our project worked with applying different methods of regression on the datasets generated by the Franke function - here are the various results for pol. deg. 5:

3.1.1 Ordinary Least Squares

First of, we look at MSE results for scaled non-noisy and noisy in addition to noisy non-scaled.



(a) No added noise (b) Noise added (c) Noisy not scaled

Figure 1: MSE values for OLS

And as we can see, there is no difference between the noisy scaled and non-scaled data - which reinforces my decision to forgo scaling of the Franke function part. Common for all three plots are that I used n=100 datapoints, and 5 polynomial degrees. We can see that a low value of noise (=0.1) does not have a big effect on the results. We continue on with the dataset generated with noise and without scaling. We can look at the R2 values:

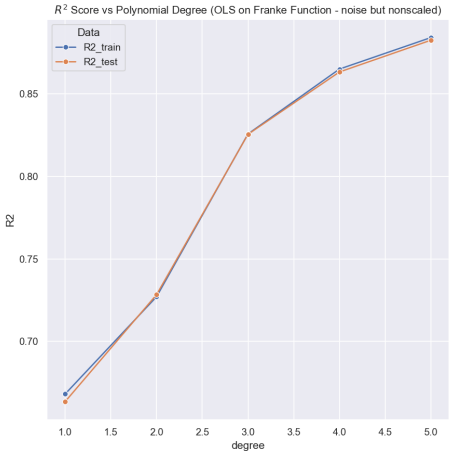


Figure 2: R2 values for nonscaled noisy OLS regression

We see that as pol. degree rises - the R2 score also becomes better - and for pol. deg. 5 we have R2 0.9 indicating a very good fit.

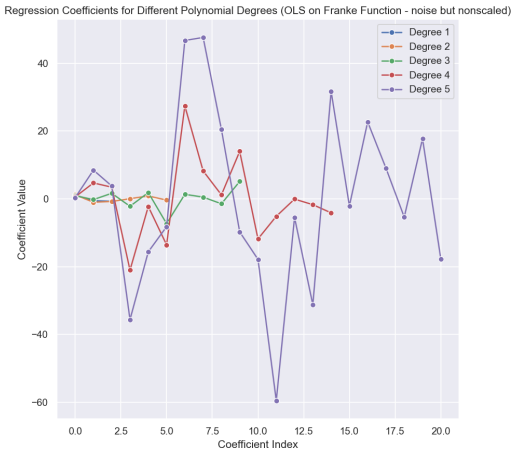
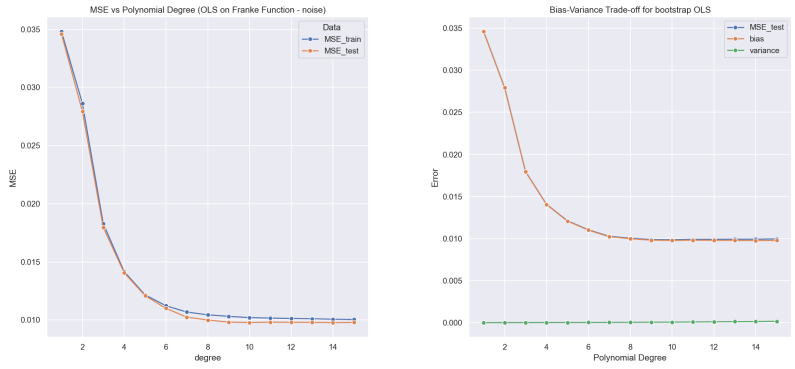


Figure 3: Beta coeffs for nonscaled noisy OLS regression

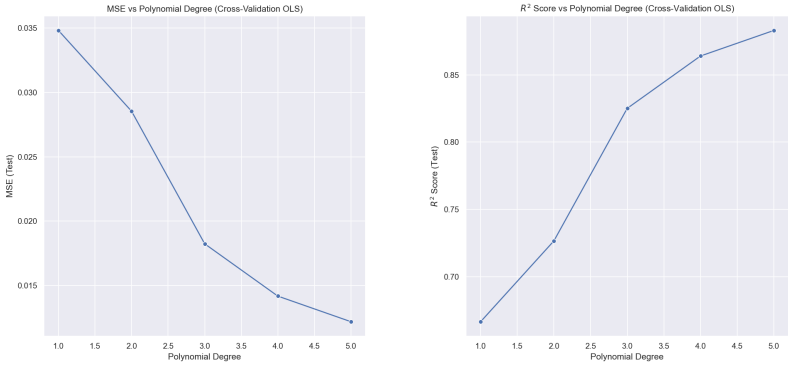
For OLS bootstrapping we have, an attempt at something akin to fig 2.11 as specified and the bootstrapping results:



(a) MSE OLS for 15 degrees (b) Boot results

Figure 4: OLS bootstrapping

And for k-fold cross validation:



(a) MSE OLS for 5-fold x-val (b) R2 OLS for 5-fold x-val

Figure 5: OLS 5-fold x-val

3.1.2 Ridge regression

For ridge regression, it might be best to just look at the heatmaps of test vs train MSE:

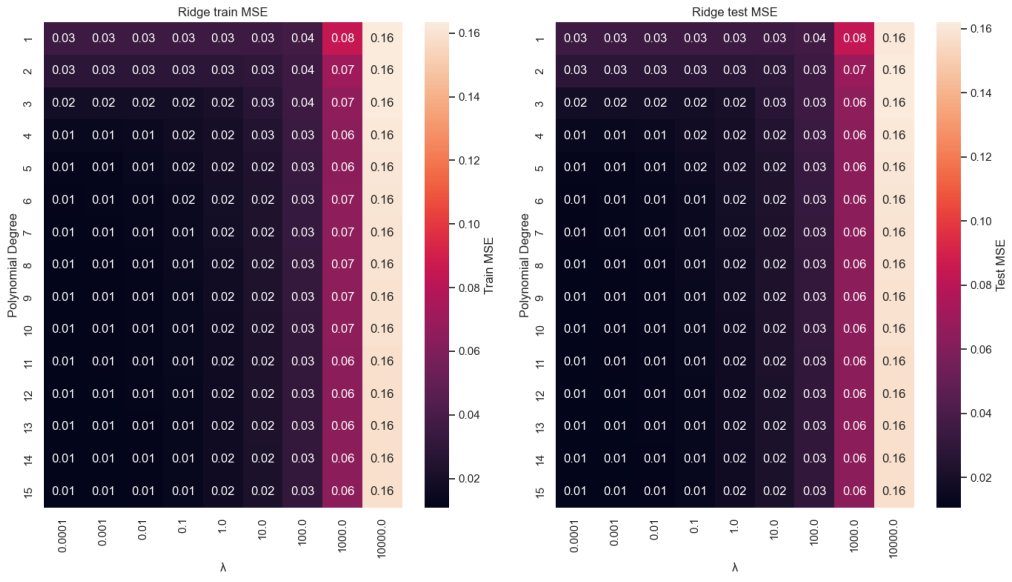
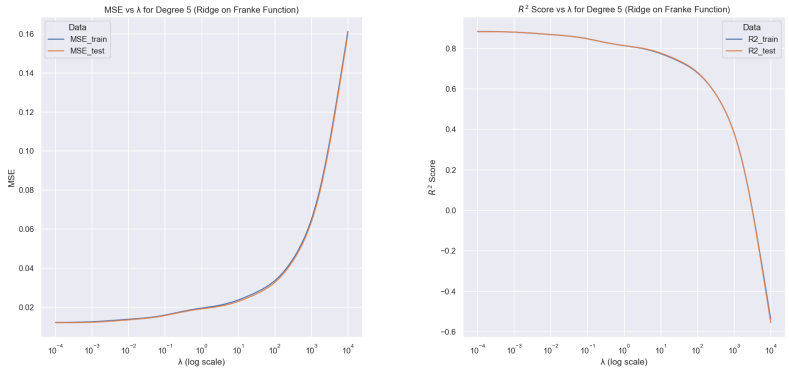


Figure 6: Ridge reg. MSE heatmaps

For ridge reg. I also have a cv MSE test heatmap:

Figure 7: Ridge reg. CV MSE test heatmap

and deg=5 metrics:



(a) Ridge MSE (b) Ridge R2

Figure 8: Ridge deg=5 metrics

3.1.3 Lasso regression

Akin to ridge regression, heatmaps is probably best for lasso as well:

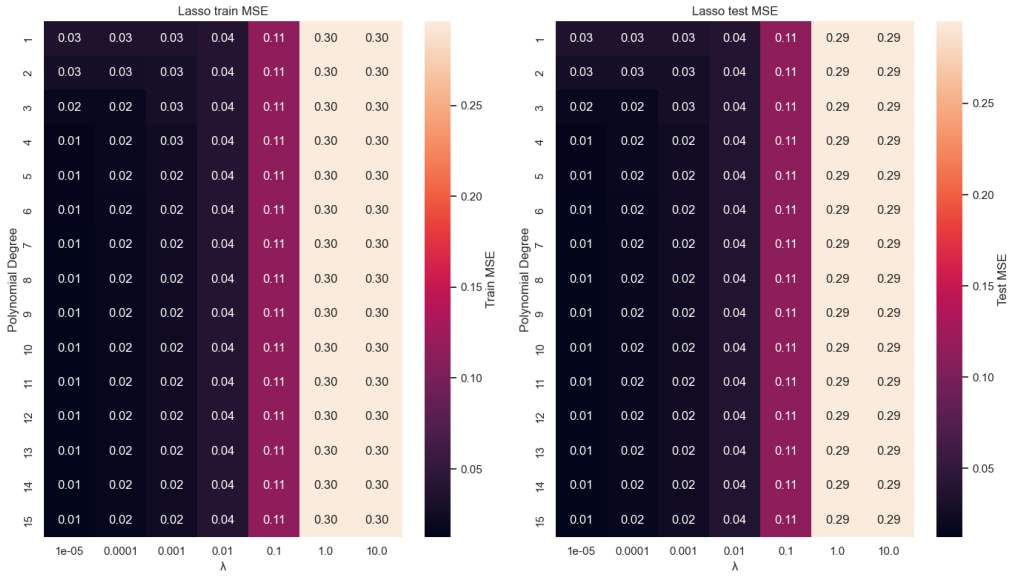
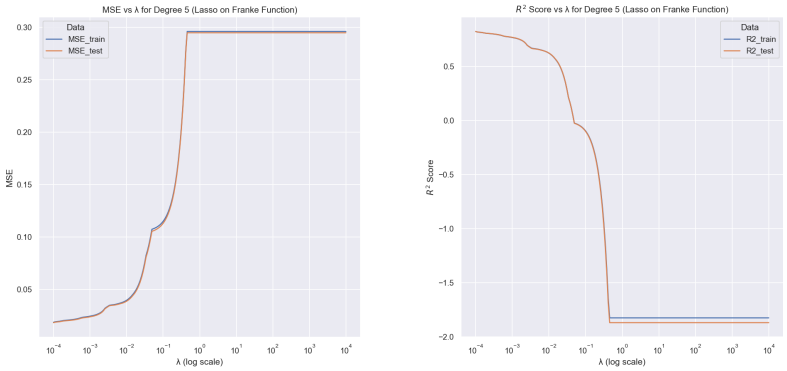


Figure 9: Lasso reg. MSE heatmaps

and deg=5 metrics:



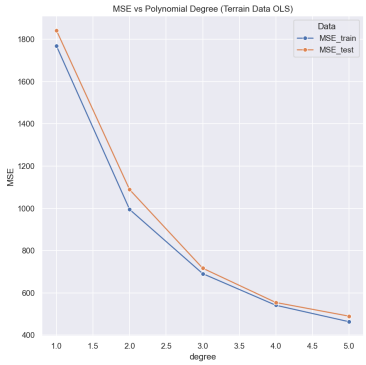
(a) Lasso MSE (b) Lasso R2

Figure 10: Lasso deg=5 metrics

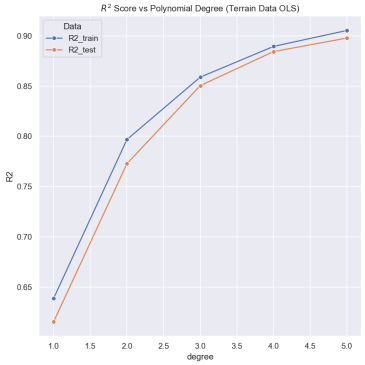
3.2 Real terrain data

3.2.1 Ordinary Least Squares

For real terrain data OLS, we can look at MSE and R2 first:



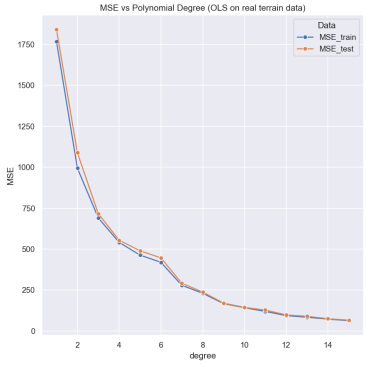
(a) OLS MSE terrain



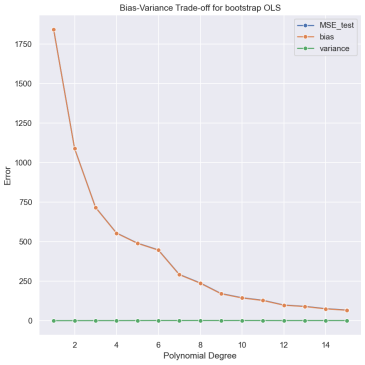
(b) OLS R2 terrain

Figure 11: OLS terrain metrics

for bootstrapping:



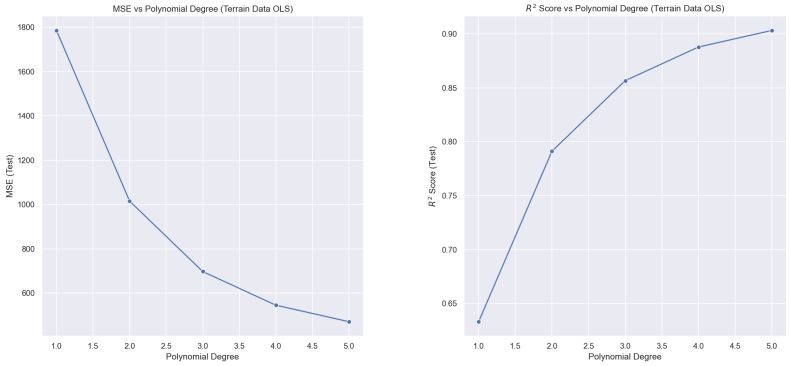
(a) OLS 15 deg MSE terrain



(b) OLS boot terrain

Figure 12: OLS bootstrapping terrain metrics

And with cv:



(a) OLS CV MSE terrain (b) OLS CV R2 terrain

Figure 13: OLS CV terrain metrics

3.2.2 Ridge regression

Heatmaps for ridge MSE's for terrain data (useless in current state):

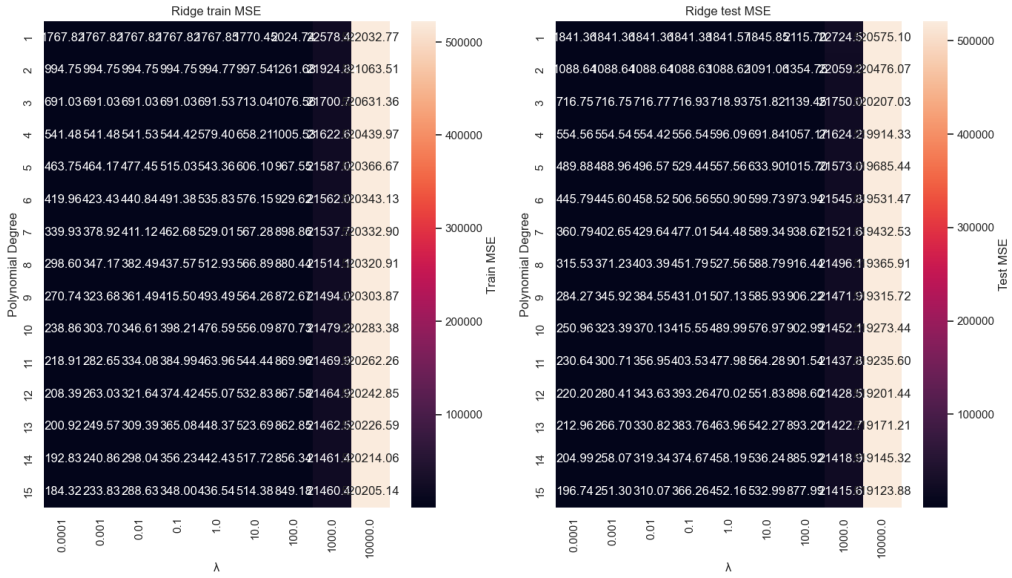
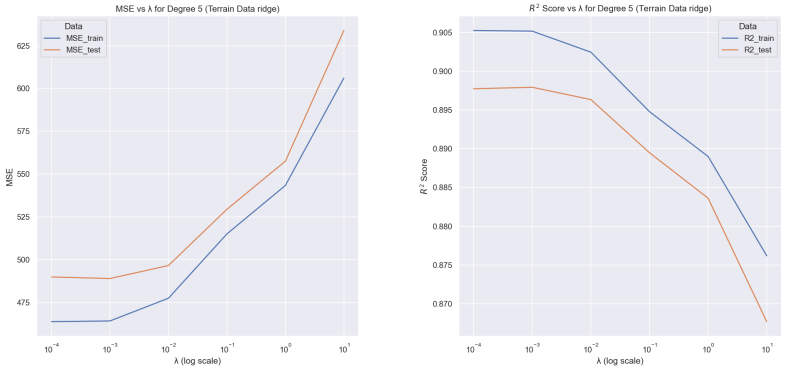


Figure 14: Ridge reg. terrain heatmap

Ridge on terrain data for degree=5

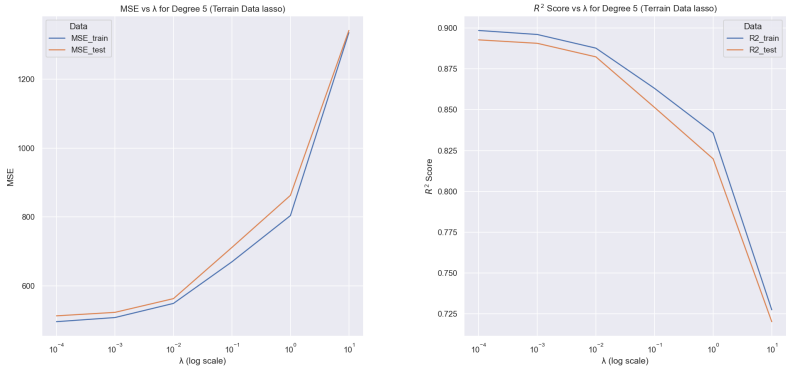


(a) ridge MSE terrain (b) ridge R2 terrain

Figure 15: ridge terrain metrics

3.2.3 Lasso regression

Lasso on terrain data for degree=5



(a) lasso MSE terrain (b) lasso R2 terrain

Figure 16: lasso terrain metrics

4 Conclusion

Looking at the results, for the Franke function generated data - all 3 models perform similarly well if when using appropriate variables for them. This means for example that for lasso to perform well, we need to include 1e-5 in the lambda range.

Due to time constraints and other personal issues, I did not manage to make everything as it should be in time - and the report and later stages of the real data analysis suffered the most. In my current results, the best performer on the real data is the OLS function.

References

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Hjorth-Jensen, Morten (2021). *Applied Data Analysis and Machine Learning*. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/intro.html [Accessed: (12.10.2024)].

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

T. Hastie R. Tibshirani, J. Friedman (2009). *The Elements of Statistical Learning*. 2nd ed. Springer New York, NY. DOI: https://doi.org/10.1007/978-0-387-84858-7.

The-pandas-development-team (Feb. 2020). *pandas-dev/pandas: Pandas*. Version latest. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

Waskom, Michael L. (2021). "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60, p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

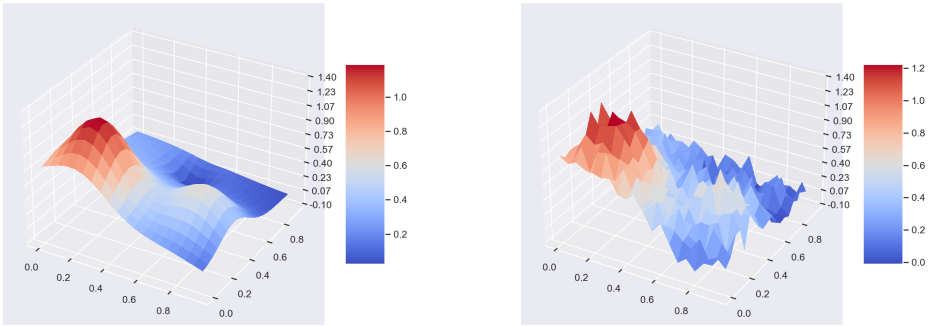
Appendix

Appendix A: Data and python libraries

In this project I use two different types of data to implement the different regression methods and resampling techniques outlined above. These methods are outlined below in A1 and A2, but both of them made use of several python libraries. Data generation and handling was done via numpy (Harris et al., 2020) while both matplotlib (Hunter, 2007) and (Waskom, 2021) were used to the different types of plots and figures. To make the implementation of some of the more complex methods easier, I used scikit-learn (Pedregosa et al., 2011), which I used for the implementation of lasso regression and k-fold cross validation. I also used pandas ((The-pandas-development-team, 2020)) for storing of the various regression metrics, this was mainly done in hopes of easing implementation of seaborn and formation of latex tables directly.

A1: Franke's function

The first parts of the project was based on the datasets (or data variables) made via the Franke function. In this project it was defined for $x, y \in [0, 1]$ - and is thus inherently scaled which means that additional scaling was not implemented the Franke funcion data. I created the x,y meshgrid using numpy's random.uniform function before also adding normally distributed noise - $N(0, 1)$. The difference can be seen in Figure 1 below.



(a) No added noise (b) Noise added

Figure 17: Data generated via the Franke function with and without noise

The data above were generated using the Franke function by following the method stated in the project task:

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

A2: Real terrain data

The real terrain data I used, was the one provided in the project folder ('SRTM_data_Norway_1.tif'), and described in the project task as being from a region close to Stavanger in Norway. Due to the size of this data (being a 3601x1801 grid), I had to sample it, and went with the same subsample size as I did for the Franke function - n=100. A 3d-illustration of the data can be seen in the figure below:

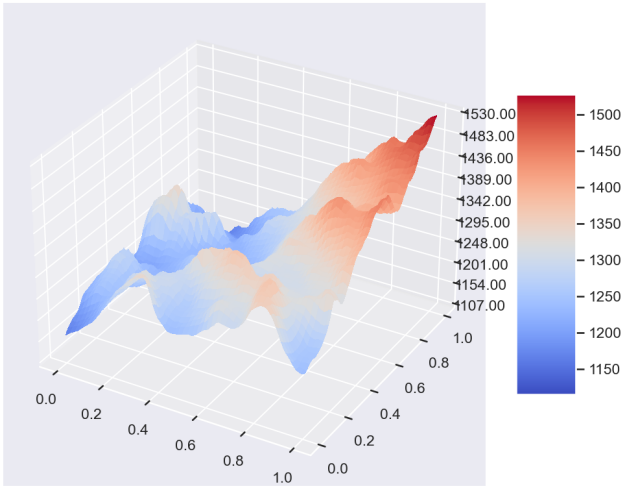


Figure 18: 3D visualization of the terrain data

Appendix B: Code and more results

All code pertaining to the generation of plots and so on can be found on my public github repository: https://github.com/Alebegur/Project-1. There is one sole .ipynb notebook that contains all code I've written and used for the project, and a result folder containing various plots from a lot of runs. Sadly, due to personal reasons and time constraints - there are a few (especially the latter parts involving lasso) parts of the code that are subpar or not working as intended.