Bias-Variance Tradeoff in Simple Linear Models

Anna Hjertvik Aasen, Carl Martin Fevang, Håkon Kvernmoen

December 4, 2022

Abstract

In this project, we explored evaluation of goodness of fit of the OLS regression method against 2D terrain-like data; first the Franke function and then on real terrain data from the Nicaraguan mountains. We used different polynomial expansions of the 2D inputs as our features, exploring the balance of the bias of low-complexity models against the variance and overfitting of high-complexity ones. To do this, we employed two resampling techniques, bootstrapping and k-fold cross-validation. Further, we looked at two regression models with an added L_p penalisation on the magnitude of the parameters to see how this affects the stability of the fit, and find that their variance is dampened as the penalisation is intensified. Finally, we compare their application on the real terrain data, and find that OLS is most easily fitted, and does best on the MSE score.

1 Introduction

In this project, we will introduce some of the most important general properties of machine learning models in the context of linear regression. Specifically we look at some popular measures for the goodness of fit of a model, assessing how well a statistical model recreates real data. We look at the phenomena of underfitting and overfitting, and explore them in the context of bias-variance tradeoff. The three models we will use explicitly are Ordinary Least Squares (OLS), Ridge regression and Least-Absolute-Shrinkage-and-Selection-Operation (Lasso) regression. These will first be tested on the 2D Franke function, and later on real geographical terrain data.

The aforementioned models will be explored using resampling techniques, which will be employed to extract the relevant statistical quantities in an accurate way. We will use the standard methods of Bootstrapping and k-fold cross-validation and compare the results we get between them across the three linear models.

All the models will be applied by fitting a two-dimensional polynomial expansion to the observed data.

2 Theory

Before we delve into to details of the theory section, it is useful to take a moment to define the quantities we will be dealing with. Let $\boldsymbol{y}$ denote a vector of a series of measured values y_i, $i \in 1, 2, \ldots, n$ at points X, where X is a matrix where the rows x_i correspond to the input values that produced the measurement y_i. For completeness, we refer to the columns of X as $\boldsymbol{x}_a$, $a = 1, 2, \ldots, p$.[1] Together, these form the dataset $\mathcal{D} = \{(y_i, x_i)\}$. Another important set of values is $\boldsymbol{\theta}$ which contains the parameters that will define the model we will use to predict the data.

We will assume that the measurements y_i are generated from an exact function $f(x)$ with stochastic noise ϵ added, such that $y_i = f(x_i) + \epsilon$. Throughout this project, we will assume that the noise to be distributed normally with zero mean and some standard deviation (std) σ; $\epsilon \overset{d}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. Furthermore, we will assume that the noise between different measurements is independent and identically distributed (i.i.d.). Our job then is to model an $\hat{f}(x)$ that we want to be close to $f(x)$.

2.1 Error Assessment and Model Optimisation

2.1.1 Measure for Goodness of Fit

When we want to create a statistically informed model that is able to predict the outcome of future measurements, we need to develop metrics for how well our model fits the data we already have. That is, given a model parametrised by parameters $\boldsymbol{\theta}$, we want to optimise them such that $\hat{f}(x) \approx f(x)$. We call this the goodness of fit. To estimate how well we achieve this with our given dataset $\mathcal{D}$, we will start with the mean squared error (MSE). Given a model which predicts values $\hat{\boldsymbol{y}}$, the mean squared error is given by

$$\mathrm{MSE}(\hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}(\boldsymbol{y} - \hat{\boldsymbol{y}})^2, \tag{1}$$

and is a strictly positive metric that we want to minimise.

Another popular metric we will use to evaluate the goodness of fit is the R^2-score, calculated as

$$\mathrm{R}^2(\hat{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{(\boldsymbol{y} - \hat{\boldsymbol{y}})^2}{(\boldsymbol{y} - \bar{y})^2}, \tag{2}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the sample mean of the observed data. This metric falls on the interval $(-\infty, 1]$, where 1 denotes the perfect fit, 0 corresponds to a constant prediction of $\hat{y}_i = \bar{y}$, and negative values corresponds to worse predictions than this.

To get further insight into the validity of our model, we can decompose the MSE into its *bias* and *variance*, which are quantities that can tell us more about exactly what our model is doing wrong. To set up this decomposition, we consider a situation where we have fitted a model to a series of L different datasets, with parameters $\hat{\boldsymbol{\theta}}^{\ell}$, with $\ell = 1, \ldots, L$, and predictions $\hat{\boldsymbol{y}}^{\ell}$ on the dataset we want to compare to. The

[1] Throughout this project, a, b will always index the p features for a model, and i, j the n data points.

sample mean squared error at each observed point can then be decomposed into

$$\mathrm{MSE}(\hat{y}_i) = \left(\mathrm{Bias}(\hat{y}_i)\right)^2 + \mathrm{Var}(\hat{y}_i), \tag{3}$$

where

$$\mathrm{Bias}(\hat{y}_i) = \frac{1}{L}\sum_{\ell=1}^{L}\left(y_i - \hat{y}_i^{\ell}\right), \tag{4a}$$

$$\mathrm{Var}(\hat{y}_i) = \frac{1}{L}\sum_{\ell=1}^{L}\left(\hat{y}_i^{\ell} - \frac{1}{L}\sum_{m=1}^{L}\hat{y}_i^{m}\right)^2, \tag{4b}$$

are the sample bias and sample variance of the model at the predicted point. This decomposition is done in more detail in Appendix A. The bias tells our model's tendency to err on a certain side of the 'true' value. The variance describes the model's spread in prediction as the model is fitted to different datasets.

2.1.2 Central Limit Theorem

The central limit theorem tells us that when sampling a series of quantities $\{z_i | i = 1, \ldots, n\}$ with independent and identical distributions with a mean μ and variance $\sigma^2 < \infty$, the distribution of the sample mean $\bar{z}_n = \frac{1}{n}\sum_{i=1}^{n} z_i$ will approach a normal distribution. Explicitly, we have

$$\lim_{n\to\infty}\sqrt{n}(\bar{z}_n - \mu) \overset{d}{\sim} \mathcal{N}\left(0, \sigma^2\right). \tag{5}$$

In practice, for some notion of a large sample size n, the approximation of the sample mean as being distributed as

$$\bar{z}_n \overset{d}{\sim} \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right) \tag{6}$$

becomes sound. Colloquially we say the sample mean approaches the true mean μ. This is powerful, as it means not only that the sample mean becomes better as the sample size grows, but quantities derived from its distribution are also determined more precisely. Note also that this holds with only very weak restrictions on the distribution of the z_is — their sample mean will approach normal distribution regardless. This lays the foundation for statistically informed models such as regression methods.

2.2 Statistical Introduction to Linear Regression

A way to motivate the process of statistical learning is that we want to develop a process by which we use observed data to inform our belief in a future outcome. Such a process can be developed from the ideas of Bayes' theorem, which we can use to inform us what confidence we should have in the parameters of a given model, given data points we have observed.

Bayes' theorem tells us that our *posteriori* confidence in the parameters $\boldsymbol{\theta}$, given the available dataset $\mathcal{D} = (\boldsymbol{y}, X)$, is given by the probability

$$P(\boldsymbol{\theta}|X, \boldsymbol{y}) = \frac{P(\boldsymbol{y}|X, \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(X, \boldsymbol{y})}, \tag{7}$$

where $P(\boldsymbol{y}|X, \boldsymbol{\theta})$ is the *likelihood* of observing the values $\boldsymbol{y}$ at points X given parameters $\boldsymbol{\theta}$, and $P(\boldsymbol{\theta})$ is our *prior* confidence in $\boldsymbol{\theta}$, before the data was seen. $P(X, \boldsymbol{y})$ plays the role of a normalisation, and can safely be ignored — either because we look for values of $\boldsymbol{\theta}$ that maximise the probability, or because we assume the probabilities $P(\boldsymbol{y}|X, \boldsymbol{\theta})$ and $P(\boldsymbol{\theta})$ which we can then properly normalise. Now we have a framework through which we can build a model by making assumptions about the probability distributions $P(\boldsymbol{y}|X, \boldsymbol{\theta})$ and $P(\boldsymbol{\theta})$ to give us a probability distribution for the parameters $\boldsymbol{\theta}$ that we can maximise.

2.2.1 Ordinary Least Squares

We define the ordinary least squares method by assuming that the data $y(x)$ that we want to make a fit to has a linear noise term that is normally distributed with zero mean. That is, the data is generated as $y(x) = f(x) + \epsilon$, where $f(x)$ is some analytic, non-stochastic function of the data input x, and $\epsilon \overset{d}{\sim} \mathcal{N}\left(0, \sigma^2\right)$ for some $\sigma^2 > 0$. Further, if we make a model fitting $y(x)$ by a $\hat{y}(x)$, we should expect the error at every data point (y_i, x_i) to be normally distributed with the same variance, i.e., independently and identically distributed (i.i.d.). This can be summarised

$$P(\boldsymbol{y}|X, \boldsymbol{\theta}) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(y_i - \hat{y}_i)^2}{2\sigma^2}}. \tag{8}$$

Furthermore, we assume the parameters $\boldsymbol{\theta}$ to be distributed uniformly, and as such, $P(\boldsymbol{\theta})$ just contributes to an overall normalisation. This means that maximising $P(\boldsymbol{y}|X, \boldsymbol{\theta})$ amounts to the same as maximising $P(\boldsymbol{\theta}|X, \boldsymbol{y})$, and can be done analytically.

Knowing that the Gaussian distribution has a single extremum that is a maximum, we can find the values for $\boldsymbol{\theta}$ maximising the probability from

$$\begin{aligned} \frac{\mathrm{d}P(\boldsymbol{y}|X, \boldsymbol{\theta})}{\mathrm{d}\theta_k} &= 0 \\ \Rightarrow \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\frac{1}{\sigma^2}X_{ib}(y_i - X_{ia}\theta_a)e^{-\frac{(y_i - X_{ia}\theta_a)^2}{2\sigma^2}} &= 0, \end{aligned} \tag{9}$$

which as we can see, is maximisable point by point. This means that the optimal parameters $\hat{\boldsymbol{\theta}}_{\mathrm{OLS}}$ that maximise $P(\boldsymbol{y}|X, \boldsymbol{\theta})$ are given analytically by

$$\boxed{\hat{\boldsymbol{\theta}}_{\mathrm{OLS}} = \left(X^T X\right)^{-1} X^T \boldsymbol{y}.} \tag{10}$$

An exploration of the statistical expectation and variance of the OLS results is done in Appendix B.

2.2.2 Ridge Regression

So far, we have assumed that the parameters $\boldsymbol{\theta}$ are uniformly distributed, and as such have no *bias* towards any particular value. In practice, this means that the OLS model will contort to fit itself to all values in the data. As such, a motivation for adding a bias could be for the sake of stability in the fit. These methods are called *regularised* methods. An example is Ridge regression, where one assumes that the parameters are normally distributed, such that parameter values far from the mean are thought less likely to occur. This in practice means that the model is less willing to contort to outliers in the dataset, trading it for a bias towards certain parameter values. Assuming the probability distribution for $\theta_a \overset{d}{\sim} \mathcal{N}\left(0, \tau^2\right)$,

$$P(\boldsymbol{\theta}) = \prod_{a=1}^{p} \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\theta_a^2}{2\tau^2}}, \tag{11}$$

with a derivative with respect to θ_b

$$\frac{\mathrm{d}P(\boldsymbol{\theta})}{\mathrm{d}\theta_b} = -\prod_{a=1}^{p} \frac{1}{\sqrt{2\pi\tau^2}} \frac{1}{\tau^2} \theta_b e^{-\frac{\theta_a^2}{2\tau^2}} = -\frac{1}{\tau^2}\theta_b P(\boldsymbol{\theta}), \tag{12}$$

the condition for maximising $P(\boldsymbol{\theta}|X, \boldsymbol{y})$ becomes

$$\begin{aligned} \frac{\mathrm{d}P(\boldsymbol{y}|X,\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} P(\boldsymbol{\theta}) + P(\boldsymbol{y}|X,\boldsymbol{\theta}) \frac{\mathrm{d}P(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} &= 0 \\ \left(\frac{1}{\sigma^2} X^T (\boldsymbol{y} - X\boldsymbol{\theta}) - \frac{1}{\tau^2}\boldsymbol{\theta}\right) P(\boldsymbol{y}|X,\boldsymbol{\theta}) P(\boldsymbol{\theta}) &= 0. \end{aligned} \tag{13}$$

Rewriting with a parameter $\lambda = \sigma^2/\tau^2$, the optimal parameters $\hat{\boldsymbol{\theta}}_{\text{ridge}}$ are given by

$$\boxed{\hat{\boldsymbol{\theta}}_{\text{ridge}} = \left(X^T X + \lambda\,\mathbb{I}\right)^{-1} X^T \boldsymbol{y}.} \tag{14}$$

This has the added bonus of ensuring that $X^T X + \lambda\,\mathbb{I}$ is always invertible, as $\det(\lambda\,\mathbb{I}) \neq 0$ for $\lambda > 0$.

2.2.3 Lasso Regression

Lasso regression builds on the same ideas as Ridge regression, but rather assumes parameters $\boldsymbol{\theta}$ are distributed according to the Laplace distribution $L(0, \tau)$. This looks like

$$P(\boldsymbol{\theta}) = \prod_{a=1}^{p} \frac{1}{2\tau} e^{-\frac{|\theta_a|}{\tau}}, \tag{15}$$

which as we can see is not differentiable at $\theta_a = 0$. This means that the optimisation of the probability $P(\boldsymbol{\theta}|X, \boldsymbol{y})$ is not necessarily possible analytically without putting bounds on the data $X, \boldsymbol{y}$, and even numerical algorithms like gradient descent will encounter trouble. Moreover, it is not optimisable point by point, unless bounds like $X^T X$ being diagonal are enforced. This makes Lasso generally more cumbersome. Nevertheless, we can coax a cost function out of the probability, which is minimalisable. Letting the cost function be written $C_{\text{lasso}}(\boldsymbol{\theta}) = -A \log\left(P(\boldsymbol{\theta}|X, \boldsymbol{y})\right) + B$ we can choose the constants A, B such that

$$\boxed{C_{\text{lasso}}(\boldsymbol{\theta}) = \|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1,} \tag{16}$$

where $\lambda = \frac{2\sigma^2}{\tau}$ and $\|\boldsymbol{v}\|_p = \left(\sum_k |v_k|^p\right)^{1/p}$ denotes the L_p norm of the vector $\boldsymbol{v} \in \mathbb{R}^k$.

3 Method

The code used to performed the following analysis can be found at https://github.com/hkve/FYS-STK4155/tree/main/Project1. We will initially employ OLS on the Franke function, before moving on to using resampling techniques to explore the goodness of fit. We will then move on to the regularised regression methods, Ridge and Lasso, comparing their application on the Franke function with that of OLS. Finally, we use what we have learned to compare the three methods on real terrain data.

3.1 Datasets

First, we should introduce the datasets we will be working with.

3.1.1 Franke Function

For the initial testing of our implemented fitting routines, we will use the so called *Franke function*. Being able to control the number of sample points and noise levels will serve as a good examination of our models advantages and disadvantaged. Defined for $x, y \in [0, 1]$

$$F(x, y) = f_1(x, y) + f_2(x, y) + f_3(x, y) + f_4(x, y) + \epsilon \tag{17}$$

Where the $f_i(x, y)$ are Gaussian functions, constructed to mimic the shape of a hill top landscape. More concretely, these are defined as:

$$\begin{aligned} f_1(x, y) &= \frac{3}{4} \exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ f_2(x, y) &= \frac{3}{4} \exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ f_3(x, y) &= \frac{1}{2} \exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ f_4(x, y) &= -\frac{1}{5} \exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned}$$

And ϵ is some stochastic noise $\epsilon \overset{d}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. By visual inspection, we can fine tune σ to create data that looks like noisy digital terrain data. Unless specified otherwise all fits to the Franke function will use $n = 600$ uniformly distributed points $x, y \overset{d}{\sim} \mathcal{U}(0, 1)$ with a noise level of $\sigma = 0.1$. We discuss this in Appendix C.

3.1.2 Terrain Data

We chose to sample terrain data from a mountainous region in northwestern Nicaragua, close to the border to Honduras, shown in Figure 1. The terrain data consists of a 300 by 300 point grid with one arcsecond (") resolution. When doing our fits, we restrict our data to $n = 600$, as in the earlier case of the Franke function. The points are sampled uniformly from the grid, without replacement.

3.2 Data Preparation

When fitting our models to the data, we base our models on simple polynomial expansions of the arguments of the function generating the data. We are fitting scalar function with

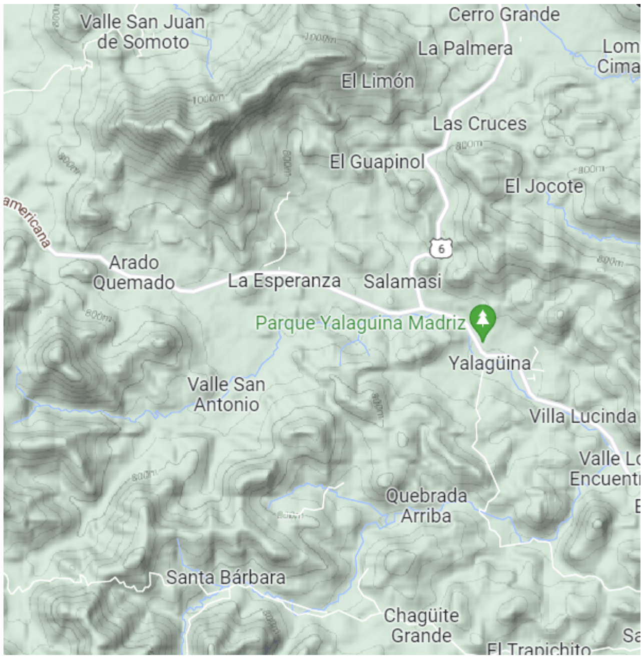


Figure 1: Map of the terrain taken from google.com/maps.
Coordinates of south-west corner: $13^\circ\,26'\,40''\,\mathrm{N}, 86^\circ\,33'\,20''\,\mathrm{W}$
Coordinates of north-east corner: $13^\circ\,31'\,40''\,\mathrm{N}, 86^\circ\,28'\,20''\,\mathrm{W}$

two arguments x and y, so our design matrix is constructed as

$$X = \begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 & \cdots & x_1^p & \cdots & y_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & y_n & x_n^n & x_n y_n & y_n^2 & \cdots & x_n^p & \cdots & y_n^p \end{pmatrix} \tag{18}$$

Before fitting the models to the sampled data, we will scale the data using the Z-score normalisation. This scales the inputs x_i and the targets $\boldsymbol{y}$ such that they have a mean value of 0 with a standard deviation 1. In effect, we rescale

$$\boldsymbol{y} \to \frac{\boldsymbol{y} - \bar{y}}{\bar{\sigma}_{\boldsymbol{y}}}, \tag{19a}$$

$$\boldsymbol{x}_a \to \frac{\boldsymbol{x}_a - \bar{x}_a}{\bar{\sigma}_{\boldsymbol{x}_a}}, \tag{19b}$$

where $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_n$ denotes the sample mean of $\boldsymbol{z}$ and $\sigma_{\boldsymbol{z}}^2 = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2$ denotes the sample variance of $\boldsymbol{z}$. When we calculate MSE scores to compare between models and resampling techniques, we will use the scaled $\boldsymbol{y}$. Unless otherwise specified, the inputs will always be scaled in this manner.

To get a proper assessment of our model, we will split the dataset into a *training set* and *testing set*. The training set is used to fit the model, giving our coefficient estimates $\hat{\boldsymbol{\theta}}$. Using the fitted model and the training data, we can calculate metrics to quantify how good our model replicates this data. In the case that our model perfectly predicts the training data $\boldsymbol{y}$ based on X, the calculated metrics MSE and R^2 (Eqs. 1, 2) should be 0 and 1 respectively. However, using these scores to assess the goodness of our fit has an underlying problem. Since the model has seen the training data to estimate $\hat{\boldsymbol{\theta}}$, we can run into problems where the model fits the ϵ values and not the (assumed) underlying function $f(x)$. This phenomenon is called *overfitting*. To mediate this, we calculate our metrics based on a testing data which was not used for fitting. This serves as a way to estimate the errors on data which the model has not seen before. For the artificial data made using the Franke function, we went for a split where $^3\!/\!_4$ of the data is used for the training set, while other $^1\!/\!_4$ is reserved for the testing set.

3.3 Initial Application of OLS

At first, we try out the OLS algorithm on our Franke data with $n = 600$ data points. We plot the MSE and R^2 of the testing and training data as we crank up the maximum degree of our polynomials in Eq. (18) from $p = 1$–12. The way the coefficient values change as complexity increases can give us insight to the stability of our model. To get a better feel for this we plot the coefficient values θ_{ab} corresponding to $x^a y^b$ for $p = 1, 3, 5$. Likewise, we tabulate the values of the coefficients for $p = 1$–5.

3.4 Resampling Techniques

With the intent to compute more accurate measures for goodness of fit (such as MSE and R^2) to evaluate our model, we artificially increase the data set by repeatedly fitting the model on subsamples of our data. This approach is called resampling and consists of a variety of methods. In this project we have applied *bootstrapping* and *k-fold cross-validation*.

3.4.1 Bootstrap

Bootstrapping consists of generating new training data by sampling data points from the original training data with replacement; meaning the same data points can be selected more than once. The new set of training data is called a bootstrapped dataset and is used to train a model. Then the relevant measures for goodness of fit are calculated. This process of resampling and calculating measures is repeated a certain number of times (bootstrap rounds), giving a distribution of the measures. The mean of the measures is then calculated across the bootstrapped models. Through the central limit theorem this mean is assumed to be the more accurate measure for goodness of fit and therefore a better evaluation of the model in question. Bootstrapping is also a method of weighing some data points more than others, which might shed new light on the data.

Together with the MSE, the bias and variance of the model are in our case relevant measures for goodness of fit. By implementing bootstrapping we wish to study the bias-variance tradeoff. Given a model of a certain complexity, we are interested in calculating these measures, not for each bootstrapped model, but across them. Given L number of bootstrap rounds (and corresponding bootstrapped models), we want the bias and variance between all the bootstrapped models at each point compared to the testing data (see Eqs. (4)). This will create a bias value and variance value for each of the input data points. When evaluating a model of a certain polynomial degree, we found an appropriate measure of the bias and variance for the *entire* model would be the mean across data points.

When employing bootstrap it is important to find an appropriate number of rounds to preform the resampling. There is a tradeoff between getting an accurate distribution of the values and keeping the process computationally feasible. To find the optimal number of rounds we perform bootstrap on the Franke data for a variety of rounds while calculating the MSE values. In this case we implement polynomial degree 7 and bootstrap rounds at constant intervals from 30–1000. There will be a distribution of MSE values for each number of rounds. By taking the mean of each distribution we can plot it as a function of rounds and find where it stabilises. One can argue that any number of rounds above this point will be excessive. We will also for each number of rounds plot the standard deviation of the corresponding MSE distribution divided by $\sqrt{L}$.

3.4.2 Cross-Validation

Cross-validation consist of partitioning the dataset into different subsets (which we will call *folds*). By using some folds for fitting the model and others for computing metrics, we can get multiple values for the same measure by changing which folds are used for fitting and which are used for calculating measures.

In this report, we will use the famous *k-fold cross-validation* technique. Considering the dataset $\mathcal{D} = (\boldsymbol{y}, X)$, consisting of n data points: Initially we partition $\mathcal{D}$ into k folds $\mathcal{D}_i$, $i = 1, \ldots, k$, of equal size. For each fold $\mathcal{D}_i$, a model is fitted using every fold *except* $\mathcal{D}_i$, which we denote $\mathcal{D}_{-i}$. With this model, we calculate the different statistical measures of the model on $\mathcal{D}_i$, which was not used to fit the model. When this is done for all k folds, we take the mean of the measures to get the final goodness of fit.

When preforming cross-validation, the computational cost is dependent on the number of folds k. Increasing k will presumably give us more confidence in our calculated metrics, but has the downside of being more computationally expensive. We wish to study the effect that k has on the calculated metrics for different polynomial degrees. This will be done by calculating the train and test MSE for $k \in [5, 10]$ and studying the differences between results obtained for different k values. The stability of the test MSE metric will be our main focus, trying to find the lowest value of k where increasing it does not yield drastically different results. Looking qualitatively for when the test MSE stabilises across folds will then presumably give us the lowest k value required while still giving accurate results.

3.5 OLS Revisited

As discussed in the previous section, resampling the data should yield a more precise evaluation of our model. We will therefore revisit the implementation of OLS, now employing resampling techniques.

3.5.1 Application of Bootstrapping

As before, we start by using our algorithm on the Franke data with $n = 600$ data points. For every model of a given polynomial degree (from $p = 1$–15) we now calculate the MSE for the training and testing data multiple times; after each of the 400 bootstrap rounds. We take the mean of the MSE values across rounds and plot the bootstrapped models as functions of model complexity. In addition we again take the mean of these models for each degree and plot it.

Thereafter, we plot the bias-variance decomposition of the MSE value from the testing data. Again, we implement the mean across bootstrap rounds as a measure for the model of that certain complexity.

Lastly we vary the number of data points in our set ($n = 60, n = 600, n = 1500$) and then plot the bias-variance decomposition.

3.6 Expanding to Regularised Regression

With the regularised regression methods, Ridge and Lasso, we introduce a hyperparameter λ that affects our fit. To explore this, we will study the MSE of the fits using cross-validation resampling as the polynomial degree and λ-value is changed. We will use the same polynomial degrees as when exploring OLS, and look at a wide range of $\lambda \in [10^{-9}, 10^{1}]$ with logarithmically spaced values. This effectively results in a grid search over (p,λ), where the lowest MSE value will give us the optimal set of paramters $(p_{\text{opt}}, \lambda_{\text{opt}})$. To be more confident in our optimal paramaters we will apply the resampling technique cross-validation when searching for the test MSE minimum. We choose to employ cross-validation as it is computationally less expensive than bootstrap (7 folds vs. 400 rounds). Computational costs and accuracy will decide the number of rounds/folds to apply for each (p, λ) value.

When we know λ-values relation to the polynomial degree, we will study the bias-variance tradeoff as a function of the complexity of our parametrisation of X for Ridge and Lasso using bootstrap resampling. When employing bootstrap as a resampling technique, the number of rounds applied will be 400, as with OLS. The analysis includes a decomposition and comparison with our previous OLS method. In this case we will apply an optimal λ for each polynomial degree (found through cross-validation).

We will further look at the difference between an optimal λ-value (found through cross-validation), and a suboptimal and larger choice of λ to see more precisely how it affects the goodness of fit. In this case the optimal λ will be the same across degrees (as opposed to the previous analysis); the λ which together with a certain degree yields the overall best result.

Finally, we will study the bias-variance tradeoff as a function of λ, with the polynomial degree fixed at a value which balances complexity and overfitting. The hope is that this will give insight into the effective degrees of freedom of the regularised methods.

3.7 Application on Real Data

Finally, we will apply the three regression methods on our Nicaraguan terrain data, always employing a resampling method. We carry out many of the same analyses as done earlier, starting by varying the number of degrees for a simple OLS fit, finding the optimal degree by minimising the MSE found using cross-validation. When moving to our regularised methods, the same method grid searching for hyperparameter optimisation will be used, searching for $p \in [5, 25]$ and $\lambda \in [10^{-11}, 10^{-6}]$ for Ridge, and $\lambda \in [10^{-8}, 10^{-3}]$. Using the optimal parameters for each model, a comparison between them will be performed to find which model most accurately recreates the test data. Lastly, for completeness, we plot our best fit to the terrain and qualitatively compare with the actual terrain.

4 Results

4.1 Ordinary Least Squares

4.1.1 Without Resampling

We begin by calculating the MSE and R^2 score (Eqs. 1, 2) fitted to the Franke function for different polynomial degrees. This can be seen in Figures 2 and 3 respectively. Based on the MSE calculated for the test data, the optimal degree was found to be $p_{\text{opt}} = 7$ with MSE ≈ 0.122. We observe that the train MSE decreases steadily with higher p, while the test MSE decreases until we reach p_{opt}, after which it increases. The opposite relationship is seen for the R^2 score.

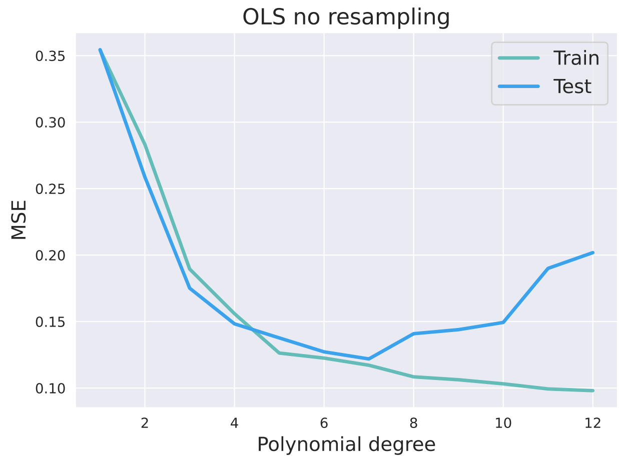


Figure 2: Showing the MSE score for OLS, calculated on the train and test sample for different polynomial degrees.

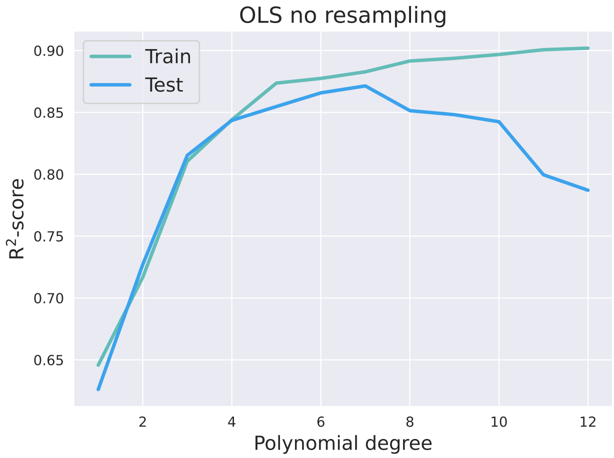


Figure 3: Showing the R^2 score for OLS, calculated on the train and test sample for different polynomial degrees.

To investigate the stability of our fits with varying polynomial degrees, we also plot coefficient values with varying polynomial degree. In Figure 4a we see these plotted for $p = 1, 2, 3$ with 95% confidence intervals. These are calculated using the diagonal $\sigma^2_{\theta_a} = \text{Var}\,(\hat{\boldsymbol{\theta}})_{aa}$. Additionally, in Figure 4b we see the values of these coefficients tabulated for $p \in [1, 5]$.

4.1.2 Exploring Resampling

The metrics thus far have been calculated without the use of any resampling techniques. To see how this influences MSE, we apply k-fold cross-validation and plot the averaged train- and test MSE across the different folds. This is presented in Figure 5. The optimal polynomial degrees and MSE values based on the training data is presented in Table 1

k	p_{opt}	MSE
5	6	0.136
7	7	0.134
10	7	0.136

Table 1: Showing the optimal degrees p_{opt} and MSE values calculate on the test data using cross-validation with $k = 5, 7$ and 10.

Just as in the case without resampling, the train MSE decreases with p. For the test MSE, the increase after the various p_{opt} values is smaller, but the general trend is the same. For further usage of cross-validation as a resampling techniquex, we use $k = 7$.

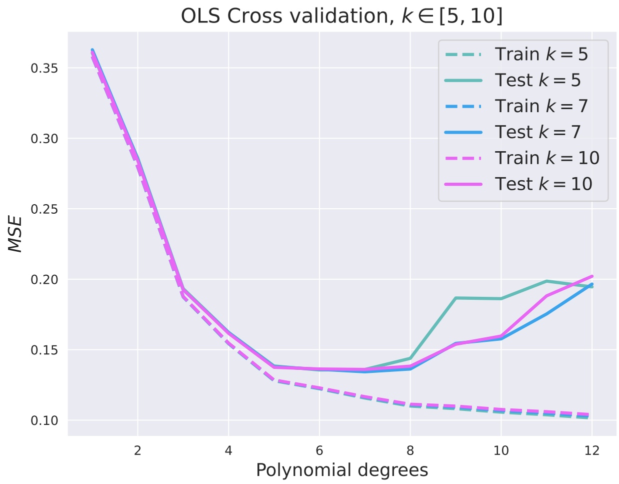
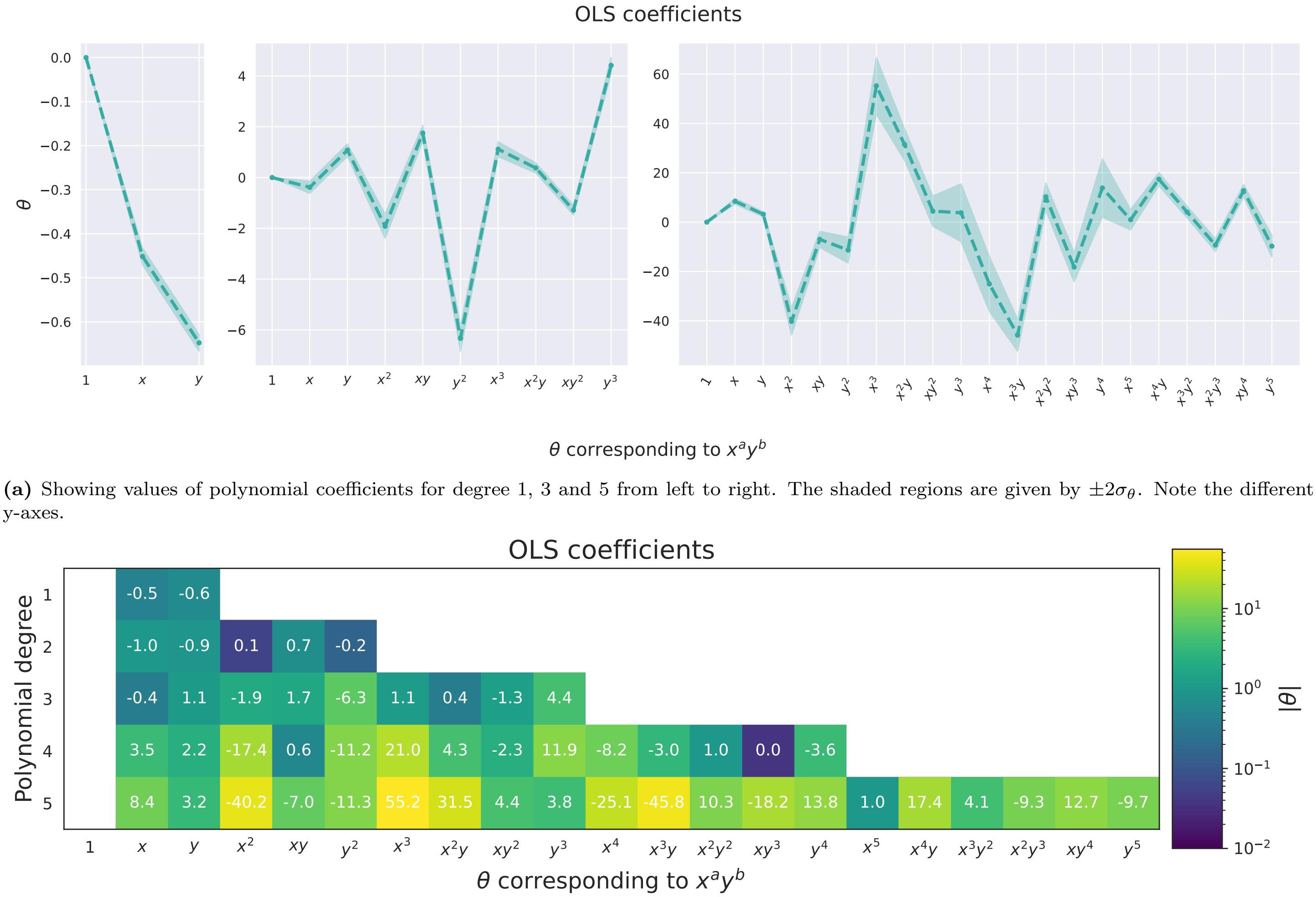


Figure 5: MSE values as a function of polynomial degrees, calculated using k-fold cross-validation for $k = 5, 7$ and 10.

When deciding the optimal number of bootstrap rounds we look at the mean of the MSE values for each number of rounds (see Figure 6). We can see from the plot that the mean MSE value varies a lot in the beginning, but stabilises around 400 rounds and remains rather constant thereafter. The same goes for the standard deviation divided by $\sqrt{L}$ (the blue field around the graph). For further exploration of the number of bootstrap rounds and the distributions of the MSE values see Appendix D.



(a) Showing values of polynomial coefficients for degree 1, 3 and 5 from left to right. The shaded regions are given by $\pm 2\sigma_\theta$. Note the different y-axes.

(b) Showing θ_{ab} corresponding to the term $x^a y^b$ present in the fit.

Figure 4: Summary of the effect on the OLS parameters as model complexity is increased.

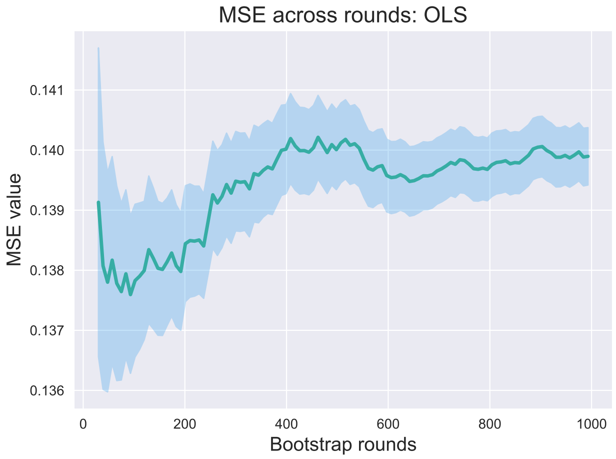


Figure 6: Plot of the MSE value as a function of number of bootstrap rounds. Each MSE value in the plot is the mean of the MSE values for that number of rounds. The blue field around the graph is the standard deviation of the MSE distribution divided by the square root of the number of bootstrap rounds. Implementing polynomial degree 7.

4.1.3 Applying Resampling

Implementing bootstrapping we can see from the plot in Figure 7 that the bootstrapped models tested on the training data yield more or less identical MSE values and decrease as the model complexity increases. However, this is not the case when employing the testing data. After the optimal polynomial degree ($p_{opt} = 6$) the MSE value not only increases (as is seen also in Figure 2), but there is a large variance in results of the bootstrapped models.

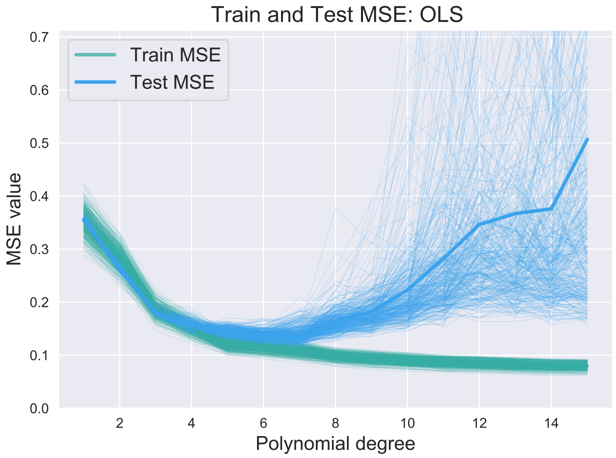


Figure 7: Plot of the MSE values calculated using the training and testing data for various bootstraps. The solid line is the mean of all the bootstrapped models.

When looking at the bias-variance decomposition of the bootstrapped MSE value (Figure 8), it is evident that when adding the decomposed parts together, it results in something very close to the original MSE, meaning the implementation is sound. The bias of the models starts high and decreases with increasing model complexity. The variance on the other hand is initially 0 at $p = 1$ and increases with the polynomial degree. The lowest MSE value occurs at $p_{opt} = 6$, for which the bias is at its lowest and variance is still small.

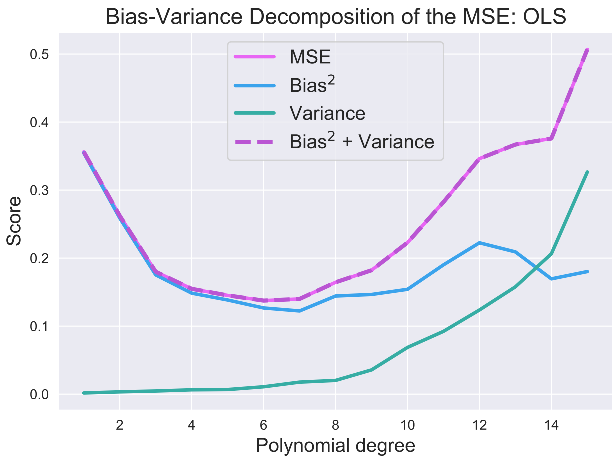
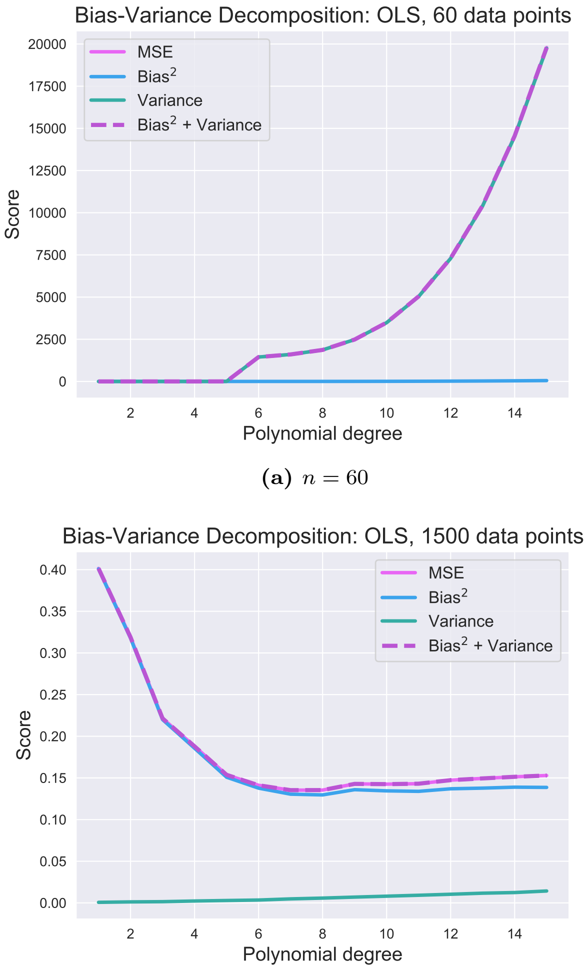


Figure 8: Plot of the bias-variance decomposition. $n = 600$



(a) $n = 60$

(b) $n = 1500$

Figure 9: Bias-variance decomposition of OLS models when varying the number of data points.

In Figure 9 we see the change in the bias-variance decomposition when varying the number of data points. For $n = 60$ the bias remains at negligible for all model complexities while the MSE and variance are identical and increase with the polynomial degree. Conversely, for $n = 1500$ the MSE and bias reach a minimum and seemingly stabilise, while the variance barely increases.

4.2 Ridge and Lasso

We repeated the same analysis for Ridge and Lasso regression, implementing resampling techniques. For cross-validation this was done for two different λ values, 0.1 and 10^{-7}, both with Ridge and Lasso regression. This can be seen in Figures 10 and 11 respectively. For $\lambda = 10^{-7}$ we again see that the test MSE stabilises when reaching $k = 7$. In the $\lambda = 0.1$ case however, there is little variation across the different k values.

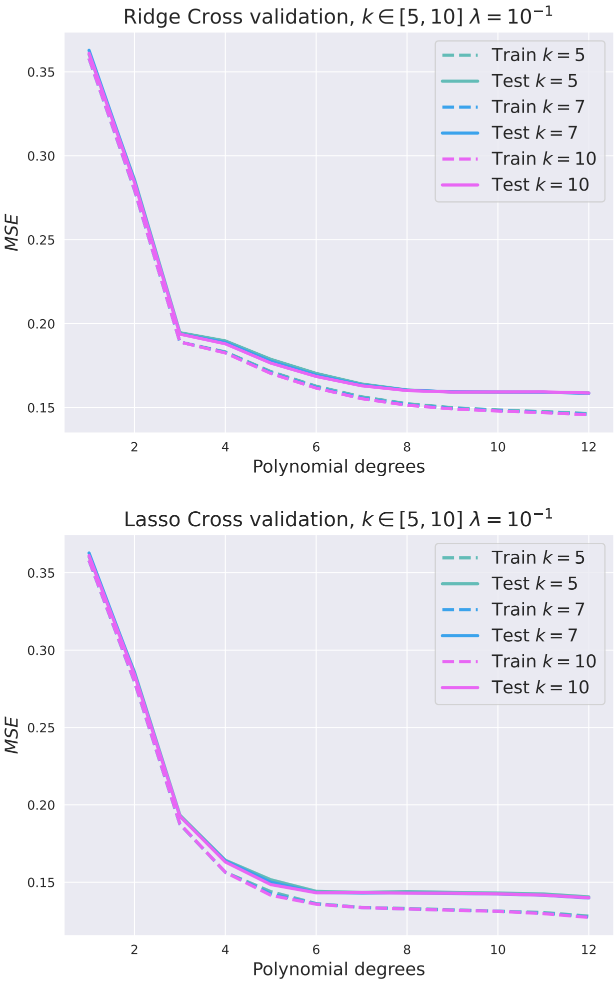


Figure 10: Test and train MSE calculated across different polynomial degrees using Ridge and Lasso regression with $\lambda = 10^{-1}$

Further, we optimised the regularised methods, grid searching for the best combination of $p \in [1, 15]$ and $\lambda \in [10^{-9}, 10^{1}]$ to find that $(p = 8, \lambda = 3 \cdot 10^{-7})$ was the optimum for Ridge and $(p = 8, \lambda = 3 \cdot 10^{-4})$ for Lasso. This is seen in Figure 14.

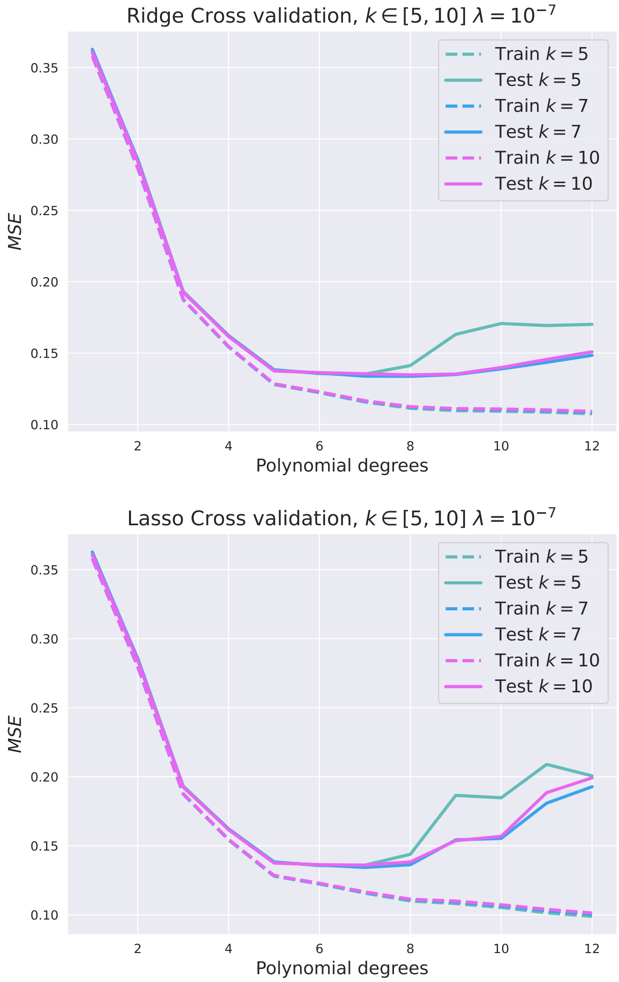


Figure 11: Test and train MSE calculated across different polynomial degrees using Ridge and Lasso regression with $\lambda = 10^{-7}$

4.2.1 Bias-Variance Decomposition

Using the optimal λ found from the cross-validation heatmap (see Figure 14) we now plot the bias-variance decomposition of Ridge and Lasso as functions of model complexity. The results can be seen in Figure 12. As opposed to OLS, where the MSE value quickly increase with polynomial degree after a certain threshold, the MSE values of Ridge and Lasso reaches a more or less stable value with only a slight increase with some fluctuation. The plots resemble the OLS for $n = 1500$. The optimal polynomial degree for Ridge is $p_{opt} = 10$ and for Lasso $p_{opt} = 9$.

We would now like to do the bias-variance decomposition for one optimal and one suboptimal λ for Ridge and Lasso. The plot of this can be found in Figure 13. To clarify, here the λs are constant across all polynomials degree. It is evident that in both approaches the suboptimal λ preforms better than the optimal λ for larger degrees (a result of the choice of optimal λ), but never reaches an MSE as low as the optimal λ.

Lastly we plot the bias-variance decomposition as a function of λ (see Figure 15) using the optimal degrees $p = 8$ for both Ridge and Lasso, taken from Figure 14.

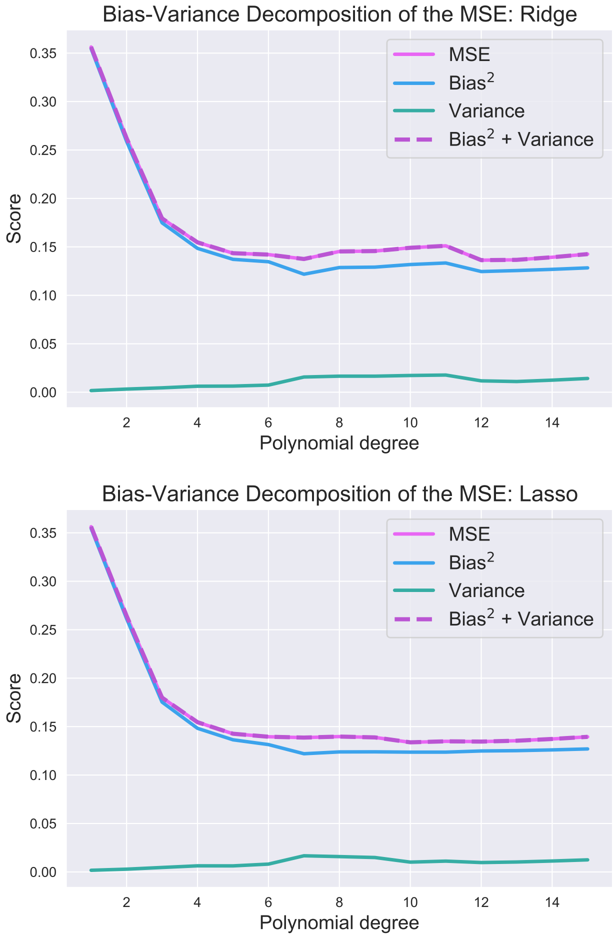


Figure 12: Bias-variance decomposition of Ridge and Lasso on Franke data with $n = 600$.

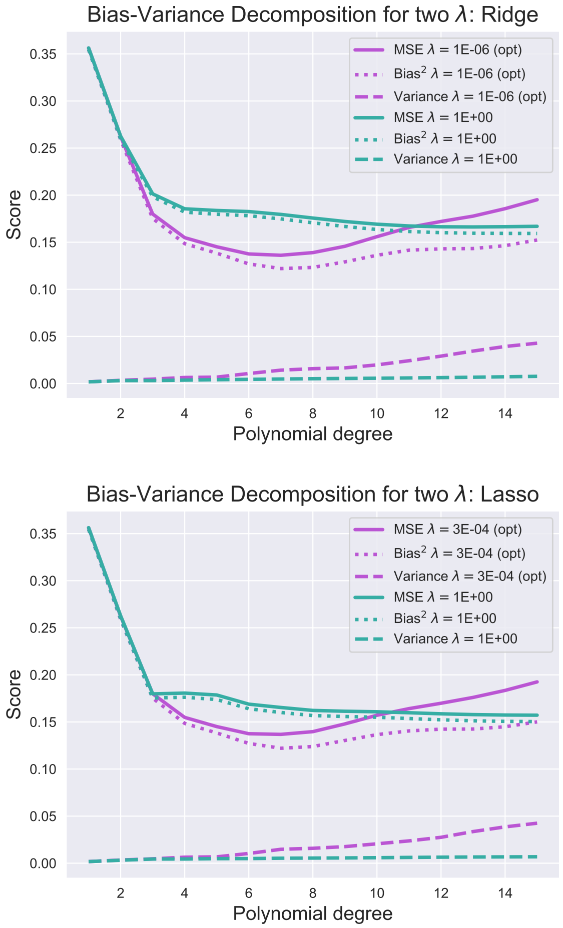


Figure 13: Bias-variance decomposition of Ridge and Lasso on Franke data with $n = 600$, now with two different λs; one optimal (found through cross-validation) and one larger.

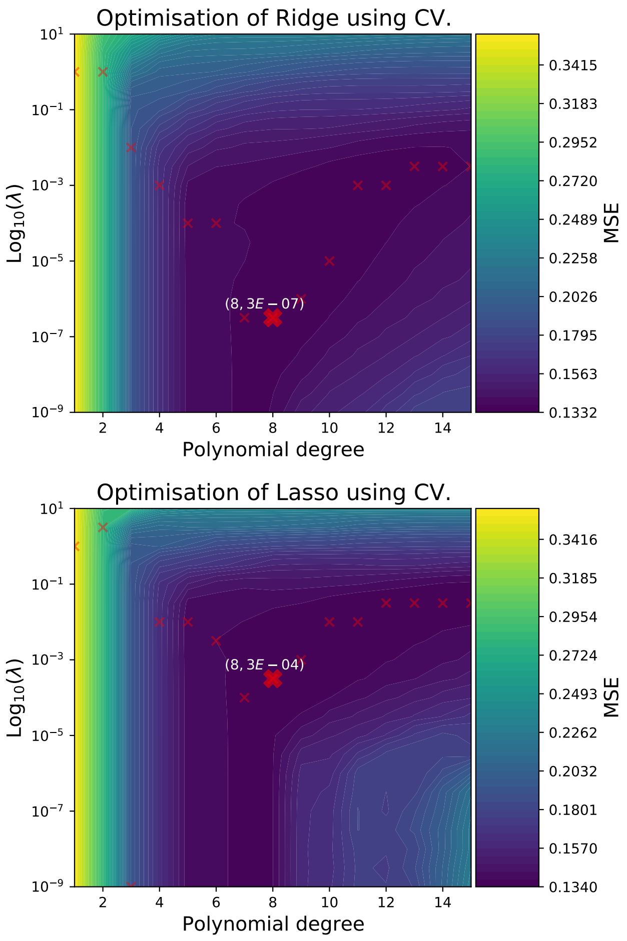


Figure 14: Grid search of the MSE of the Ridge and Lasso methods fitted to the Franke function with $k = 7$ CV resampling. We computed the MSE for every $p \in [1, 15]$ and 21 logspaced $\lambda \in [10^{-9}, 10^{1}]$. The red crosses denote the best λ-value for each degree, the large cross denoting the global minimum of the MSE.

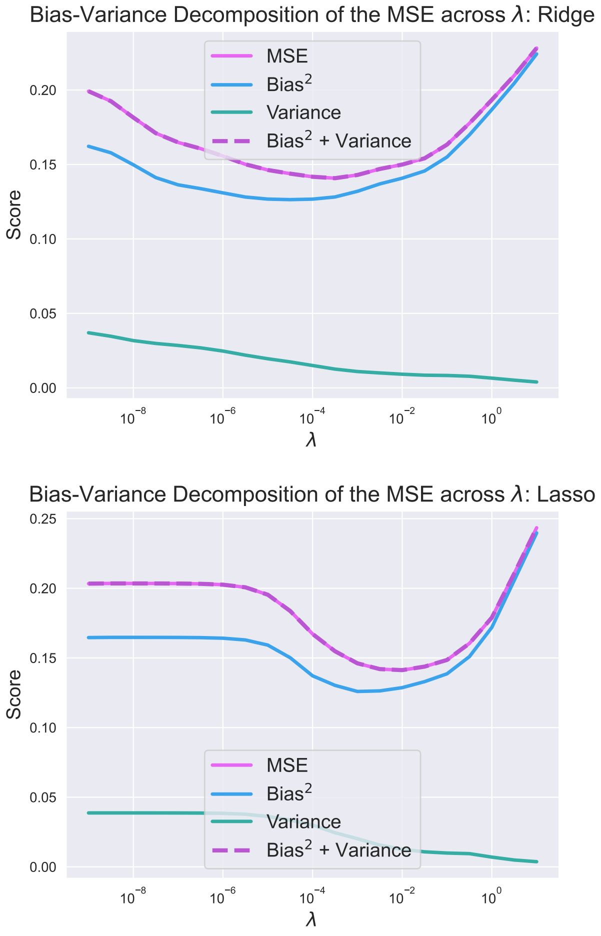


Figure 15: Bias-variance decomposition as a function of λs of Ridge and Lasso on Franke data with $n = 600$.

4.3 Nicaraguan Terrain Data

Initially, we fit the OLS method to the terrain data with $p \in [5, 25]$; this can be seen in Figure 16. From this, we conclude that $p = 12$ is the optimal degree for OLS with $n = 600$ data points on this terrain data, where the MSE has a minimum at 0.1808. We see the recognisable signs of overfitting starting at around $p = 18$.

This leads us to optimise the regularised methods, seen in Figure 17. We find $(p = 12, \lambda = 3 \cdot 10^{-9})$ as the optimum for Ridge, and $(p = 13, \lambda = 6 \cdot 10^{-5})$ for Lasso, with respective minima MSEs 0.1865 and 0.1887 — slightly underperforming OLS.

Finally, we fit our OLS model with $p = 12$ on the entire terrain dataset, and plot the prediction against the original data in Figure 18.

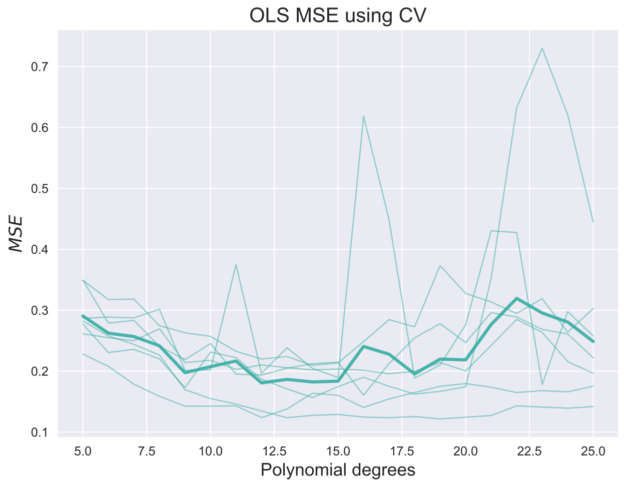


Figure 16: Optimisation of the MSE of OLS on the terrain data using $k = 7$ cross-validation. The thinner lines denote the MSE by fold, and the thicker line the mean. Minimum at $p = 12$ with $\text{MSE} = 0.1808$.

5 Discussion

5.1 Data Scaling

The scaling of the inputs and measurements can roughly be divided into two operations, each with their own set consequences. Let us first consider shifting of the measurements $\boldsymbol{y}$ and the inputs $\boldsymbol{x}_a$ by their respective means. Effectively, this eliminates a degree of freedom from our model $\hat{f}(x)$ as we set its mean value to zero, and leave out any parameter that would govern the intercept $\hat{f}(0)$. However, if we shift $\boldsymbol{y}$ to have zero mean as well, this degree of freedom becomes redundant, and we end up effectively reducing the dimensionality of the model by one. Moreover, in regularised models where some norm of the parameters $\|\boldsymbol{\theta}\|_p$ is penalised, we remove any penalisation of the choice of the intercept. This is natural to do, as the goal of these methods is to reduce the variance of the model, something which is not affected by changing the intercept.

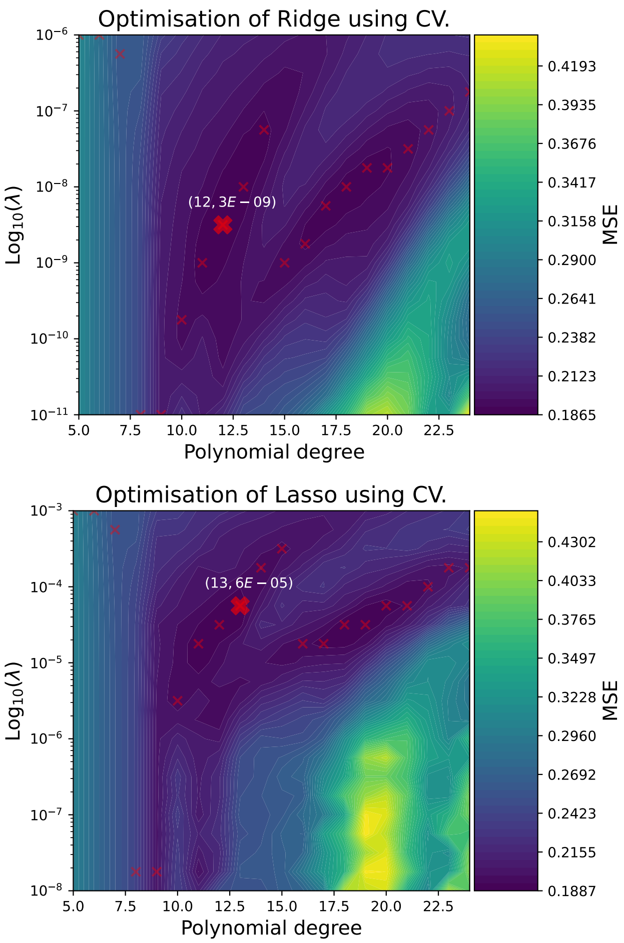


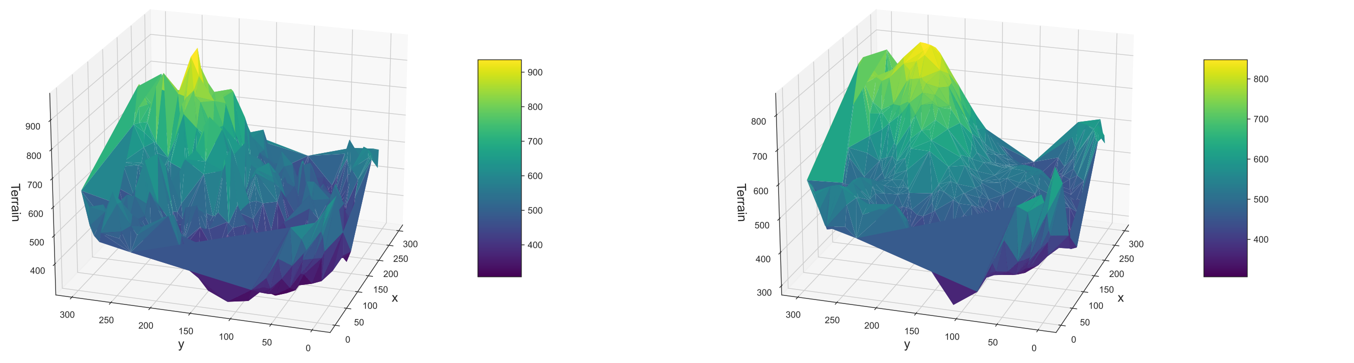
Figure 17: Grid search of the MSE of the Ridge and Lasso methods fitted to Nicaraguan terrain data with $k = 7$ CV resampling. The red crosses denote the best λ-value for each degree, the large cross denoting the global minimum of the MSE.

Considering now the rescaling of $\boldsymbol{y}$ and $\boldsymbol{x}_a$, these affect the fitting of the model more subtly. Looking at it through the lens of dimensionality know that

$$[\theta_a] = [y]/[x_a]. \tag{20}$$

From this we can infer that the order of magnitude of a parameter θ_a is governed by the dimension of its corresponding feature $[x_a]$. This matters in the regularised models where a cost is associated with the norm of the parameters $\|\boldsymbol{\theta}\|_p$, which will be dominated by parameters whose dimension is large. In practice, this means that the fitting will focus more on reducing the magnitude of the parameters whose associated dimension is large, letting the others fluctuate more freely. By making the x_a's dimensionless, the parameters are put on a 'level playing field', so to speak. The dimension of y serves to raise or lower the order of magnitude of the all the parameters equally, and can be adjusted as to avoid numerical truncation errors by avoiding larger floats.

From this, we can conclude that scaling of the inputs is not strictly necessary when using OLS, as there is no regularisation penalty associated with the dimension of the parameters. However, the MSE is still a dimensionful quantity, and it is useful then to scale the data for OLS too for consistency and ease of comparison of the results between the models.



(a) $n = 600$ randomly sampled points from our Nicaraguan terrain data. **(b)** Best fit to the Nicaraguan terrain data found using OLS with $p = 12$.

Figure 18: Comparison of the actual Nicaraguan terrain data and the unscaled prediction of our best scoring model. x denotes arcseconds north, y denotes arcseconds west. z units can be disregarded, but scale serves to allow comparison of magnitude between the plots.

5.2 OLS

Looking back at Figures 2 and 3 showing the MSE and R^2 scores calculated using the train and test set, we recognise a common sign of overfitting when moving past $p_{\text{opt}} = 5$. Since the polynomial degree increases, we can make many more $x^a y^b$ combination, which in turn gives more coefficients θ_{ab} to determine. This introduces many new degrees of freedom to the fit, which allows our model to accurately recreate the exact points present in the training set, but at the cost of fitting the added noise ϵ and/or assuming a too complex structure to the underlying function $f(x)$. This can be seen by the large MSE values calculated on the test data, telling us that our model performs poorly on data which it has not yet been exposed to. In Figure 4 we see the coefficient sizes across degrees $1, 3$ and 5 (Figure 4a) and 1–5 (Figure 4b). It is clear that when we allow for higher polynomial terms (increasing the complexity of our fit), the general size of θ_{ab} increases. We interpret this as different coefficients working together to produces the overall best fit to the data. That is, if for instance turn on the $p = 5$ terms, many of the $p = 4$ terms might need to be changed (rather much) to allow for minimising the sum of the squared residuals. The $\pm 2\sigma_\theta$ confidence level also increases for θ coefficients belonging to higher p, indicating that there is more room to tune optimal parameters for the model.

5.3 Resampling

Resampling a given data set seems on the face of it to be truly the work of magic. Somehow we invoke the law of large samples from the central limit theorem (5) that our statistical quantities, like the mean value, approaches the true value as the sample size grows. However, that is not exactly what we are doing when we are resampling. To illustrate this, we can think of a coin-flipping experiment where are results are categorically heads or tails. If we perform the experiment an *odd* number of times, we must necessarily end up with either more heads or more tails. If we then continuously resample from this dataset, a mean value should, by the central limit theorem, rather approach the *biased* mean value of the original dataset.

The value of resampling rather comes from creating a way to emphasise different parts of the original dataset when fitting our model, and consequently comparing what this does to the predictions of the model. For instance, an overfitted model might contort to accommodate a statistical outlier in the original dataset, and thereby make a poor prediction on new data. However, when resampling, many of the new datasets sampled to make new fits from our model will not include this statistical outlier, and should make predictions not influenced too heavily by this. There should then be a medium where the particular ‘oddities’ in the original dataset can be smoothed over, and the bias of the original dataset is not brought out to the front.

5.3.1 Cross-Validation and Bootstrap

Looking back at fold varying test- and train MSEs for our regularised methods, we see that the low $\lambda = 10^{-7}$ graphs (Figure 11) closely resembles the results obtained for OLS (Figure 5). This is in agreement with regaining OLS for $\lambda \to 0$. For the more regularised $\lambda = 0.1$ graph (Figure 10), we observe almost no variation between different k values. This might be due to θ-parameter space being smaller, since we add a large penalty on our coefficients. Since many of the same data points are used for fitting across different k values and the penalty term gives a large constraint on the possible θ values, the fitted models across different folds will be more similar.

The choice of $k = 7$ used for our parameter grid search is influenced by the computational aspect of the approach. Since a model has to be fitted k times for each set of parameter (p, λ), it is clear that this increases the time taken to search the parameter space. However, for both OLS and Ridge, the closed form expressions (Eqs. (10) and (14) respectively) only require matrix multiplication and inversion. Adding more folds could potentially be feasible here. However, since Lasso requires a cost function minimization, it is significantly slower than OLS and Ridge, especially for high p. Therefore, we get a tradeoff between the number of (p, λ) to try and the number of folds k, where $k = 7$ was found to allow for high parameter resolution while still being completed in a feasible amount of time. As discussed, if bootstrapping would be applied for the grid search, the computational costs would be even worse since we would need to fit $L = 400$ models for each (p, λ).

The bootstrap holds the test set constant during calculation of metrics. This is in contrast to cross-validation where the test set is different (with no overlapping points) for each fitted model. Holding the test set constant might be beneficial since every model calculates metrics based on the same data, thus averaging over MSE scores gives a more accurate picture of the MSE present in that specific test set. However, this introduces bias since we assume that the selected test set accurately represents the population, which it may not. Cross-validation on the other hand does not average MSE values over the same test data and larger variations may occur. The advantage of this is that metrics are calculated on the entire data set, not just a specific partition.

When extracting the optimal hyperparameter, λ_{opt}, from the cross-validation heatmaps it has a corresponding optimal polynomial degree , p_{opt}. However, we notice that when implementing that λ_{opt} in the bias variance decomposition using bootstrap as the resampling technique, it yields a different p_{opt}. This have multiple plausible explainations. As the bootstrapped MSE value more or less stabilises after a certain degree and then fluctuates (see Figure 12), the fluctuations might indicate exactly which polynomial degree yields the best results and it might therefore differ from cross-validation. Furthermore, in our implementation of cross-validation we end up changing the amount of training data we have. With 7-fold cross validation, only $^1/_7$ of the data remains as testing data, while in bootstrapping this amount is $^1/_4$. This will also have an impact on the MSE values themselves.

5.3.2 Bias Variance Tradeoff

The bias-variance decomposition is a method for further analysing the MSE value of a model and find out what makes the model preform well/badly. As the model approximates the training data (the complexity increases), it gets closer to capturing the true relationship between the input and the output; the bias of the model decreases. Simultaneously the model gets more dependent on exactly which data it is trained on; the variance increases. An ideal model will have a low bias and a low variance, but it is rare that these coincide and one is forced to find a tradeoff.

We analyse bias variance tradeoff through decomposition of the MSE. For OLS we see the decomposition visualised in multiple plots. In the plots in Figures 8 and 7 the change in variance coincides; the difference between the various bootstrapped models increase with model complexity, and this is what we see as variance in the bias-variance decomposition. Further in Figure 8 the model yielding the lowest MSE value is given by ($p_{opt} = 6$), where the bias is at its lowest and the variance has not yet increased much. This is an example of the bias-variance tradeoff.

When looking at the bias variance tradeoff for various n we see that the bias remains low for small numbers of data points, which corresponds with the notion that little complexity is required to approximate few data points. It also makes sense that the variance and MSE in that case quickly increase with the model complexity; the model is easily overfitted. For $n = 1500$ the number of data points impedes the model from overfitting at such low polynomial degrees. Had we further increased the model complexity, we should again see the signs of overfitting.

In the bias variance decomposition of Ridge and Lasso where there is one optimal λ per polynomial degree (see Figure 12) there is little to no occurence of overfitting. However, in the results from the suboptimal and larger λ (see Figure 13) one can see that the lowest MSE value of the optimal λ is never reached. This is a clear sign of underfitting. We can also see that the models of optimal λs reach a minimum MSE value and thereafter increase with the model complexity. This is not the case for the models of the larger λs, but it is plausible that it would be if we were to increase the polynomial degree further.

The value of the hyperparameter, λ, in Ridge and Lasso is not directly comparable as they differ in its definition (see Eqs. (14) and (16)).

When keeping the polynomial degree fixed and varying λ in Figure 15, we saw similar results between Ridge and Lasso, mainly that the variance decreased as λ is increased, while the bias initially decreases before shooting up again. This mirrors the plots where the polynomial degree is fixed, telling us that large λ-values limit the complexity of our models, resulting in low variance and large bias. As we decreased λ, we find an optimum tradeoff between the bias and variance before they approach the OLS results.

The difference in the response to an increased λ between Ridge and Lasso is that Ridge seemingly moves more smoothly from mimicking OLS before turning the effective degrees of freedom off, whereas Lasso suddenly does so (at around $\lambda = 10^{-3}$ in our case). This difference can be leveraged depending on the situation, but as long as we optimise thoroughly, as we have done, this difference does not matter as much.

5.4 Fit to Terrain Data

First, we should take a minute to discuss the choice of terrain data to fit to, and how this could change things. We naturally chose a mountainous terrain, for there to be more interesting features for the models to recapture. However, we chose a large region, with a full resolution (300×300 points) on which we could not hope to do our full analysis, using of all the models and resampling adequately. We could either choose a smaller sub-area or sample more sparsely from the region. We chose the latter, as this would more closely resemble noisy data or a situation where we might not have all the data we would want for our regression. This highlights the power of regression methods to sift through data and find the most salient features. This is seen in Figure 18, where our fitted model recreates a smoothed version of the jagged mountains where the difference from the low valley to the high peak is obvious, but without the sharpness and details of the original data.

The different regression methods we used on the terrain data all performed similarly when optimised, with optimal MSEs of 0.1808, 0.1865 and 0.1887 respectively for OLS, Ridge and Lasso. However, the window within which the OLS method works well is narrow, quickly moving from high bias to high variance, as evidenced in our earlier analyses. In Figure 17 we see that there are larger areas where Ridge and Lasso perform well, but at the cost of 'fine-tuning' both the λ-parameters and the model complexity. The regularised methods are more consistent across model complexities, but come at the cost of an additional parameter to tune; in a sense they are 'safer' to employ.

Ultimately, none of the methods do a brilliant job of reproducing the exact features of the jagged terrain we apply them on, as we do try to fit smooth functions to something inherently irregular. Had we focused on a smaller region with a larger resolution, our results would surely be better.

6 Concluding remarks

Applying linear regression methods is pretty straight forward on scalar data generated from 2D inputs. It can give good insight into an underlying function generating noisy data, as long as the function is somewhat smooth. The fundamental problem to solve when using these methods is to find a good balance between model complexity, introducing variance into the fits, and simplicity, introducing bias. We found that this was quite easily done with the OLS method on both the Franke function and on our Nicaraguan terrain data. Bootstrap resampling was used to explore these quantities, while both bootstrapping and k-fold cross-validation was used and to invoke the CLM to extract more accurate quantities to quantity the goodness of fit, like the MSE and R^2 scores. We then expanded to regularised methods like Ridge and Lasso regression, which introduce a hyperparameter which penalises overfitting models, and makes for more stable fits. Effectively, these gave us a complexity slider that allowed for more smooth variation of model complexity than discrete changes to our polynomial feature expansion. In the end, these methods were not necessary to employ on our datasets, and rather introduced another dimension to optimise to create the best possible fit to the data. Bootstrap resampling allowed us to explore the bias-variance tradeoff of these methods, whereas cross-validation was employed in the more computationally expensive optimisations of the regularised methods due to its computationally lower cost while retaining adequate results.

A Bias-Variance Decomposition of the MSE of a Linear Model

The true MSE of a model $\hat{f}(x)$ to data constructed from $f(x)+\epsilon$ can be decomposed into terms resulting from the bias of the model, the variance of the model and the *irreducible error* resulting from the error in the observations. The bias encapsulates tendency of the model $\hat{f}(x)$ to under- or overpredict the value at a point x as the data used to fit the model is varied. The variance encapsulates how much the prediction at a point x changes as the data used to fit the model is varied. The decomposition follows from the definition of the MSE. We suppress the arguments x, understanding that the expectation values are taken at the same point x.

$$\begin{aligned}
&\mathrm{MSE}(\hat{f}) = \mathbb{E}\left[(f+\epsilon-\hat{f})^2\right]\\
=&\,\mathbb{E}\left[\Big((f+\epsilon-\mathbb{E}(\hat{f}))-(\hat{f}-\mathbb{E}(\hat{f}))\Big)^2\right]\\
=&\,\mathbb{E}\left[\Big(f+\epsilon-\mathbb{E}(\hat{f})\Big)^2\right]+\mathbb{E}\left[\Big(\hat{f}-\mathbb{E}(\hat{f})\Big)^2\right]\\
&-2\,\mathbb{E}\left[(f+\epsilon-\mathbb{E}(\hat{f}))(\hat{f}-\mathbb{E}(\hat{f}))\right]\\
=&\Big(\mathbb{E}\left[f+\epsilon-\mathbb{E}(\hat{f})\right]\Big)^2+\mathbb{E}(\epsilon^2)+\mathrm{Var}(\hat{f})-2\,\mathbb{E}(\epsilon\hat{f})\\
=&\Big(\mathrm{Bias}(\hat{f})\Big)^2+\mathrm{Var}(\hat{f})+\sigma^2, \qquad\qquad (A.1)
\end{aligned}$$

where in the last line, we used that $\mathbb{E}(\epsilon\hat{f})=\mathbb{E}(\epsilon)\,\mathbb{E}(\hat{f})=0.$[2]

B Theoretical Proof of Concept of the OLS Model

Supposing that we have data generated from $y_i = f(x_i)+\epsilon$ as before, we look statistical quantities derived from the distribution of our parameters $\boldsymbol{\theta}$ inherited from $\epsilon \overset{d}{\sim} \mathcal{N}\left(0,\sigma^2\right)$. Under the assumption that $f(x)\in\mathrm{span}\left\{x_a\right\}, a\in 1,\dots,p$ of our design matrix X, we can write the data $\boldsymbol{y}=X\boldsymbol{\theta}+\boldsymbol{\epsilon}$ for some $\boldsymbol{\theta}$ which would be a 'true' optimum for our parameters.

The expectation value of a data point y_i is[3]

$$\mathbb{E}(y_i)=\mathbb{E}\left[X_{ia}\theta_a+\epsilon_i\right]=X_{ia}\theta_a+\mathbb{E}(\epsilon_i)=X_{ia}\theta_a, \qquad (B.1)$$

with its variance is

$$\begin{aligned}
\mathrm{Var}(y_i)&=\mathbb{E}\left[(y_i-\mathbb{E}\left(y_i\right))^2\right]=\mathbb{E}\left[(X_{ia}\theta_a+\epsilon_i-X_{ia}\theta_a)^2\right]\\
&=\mathbb{E}\left(\epsilon_i^2\right)=\mathrm{Var}\left(\epsilon_i\right)=\sigma^2. \qquad (B.2)
\end{aligned}$$

The expectation value of our parameters can be calculated specific to OLS as

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\theta}})&=\mathbb{E}\left[(X^TX)^{-1}X^T\boldsymbol{y}\right]=\mathbb{E}\left[(X^TX)^{-1}X^T\left[X\boldsymbol{\theta}+\boldsymbol{\epsilon}\right]\right]\\
&=\mathbb{E}\left(\boldsymbol{\theta}\right)+\left(X^TX\right)^{-1}X^T\,\mathbb{E}\left(\boldsymbol{\epsilon}\right)=\boldsymbol{\theta}, \qquad (B.3)
\end{aligned}$$

and their covariance matrix is

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\theta}})&=\mathbb{E}(\hat{\boldsymbol{\theta}}\hat{\boldsymbol{\theta}}^T)-\mathbb{E}(\hat{\boldsymbol{\theta}})\,\mathbb{E}(\hat{\boldsymbol{\theta}}^T)\\
&=\mathbb{E}\left[(X^TX)^{-1}X^T\boldsymbol{y}\boldsymbol{y}^TX(X^TX)^{-1}\right]-\boldsymbol{\theta}\boldsymbol{\theta}^T\\
&=(X^TX)^{-1}X^T\left[X\boldsymbol{\theta}\boldsymbol{\theta}^TX^T+\sigma^2\,\mathbb{I}\right]X(X^TX)^{-1}-\boldsymbol{\theta}\boldsymbol{\theta}^T\\
&=\sigma^2(X^TX)^{-1}, \qquad (B.4)
\end{aligned}$$

where we have used that $\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)=\mathbb{E}\left[(X\boldsymbol{\theta}+\boldsymbol{\epsilon})(X\boldsymbol{\theta}+\boldsymbol{\epsilon})^T\right]=X\boldsymbol{\theta}\boldsymbol{\theta}^TX^T+\sigma^2\,\mathbb{I}$.

[2] At first glance, this might seem odd, as there is noise involved in the fitting of $\hat{f}$ too. However, under the assumption that the noise is i.i.d. between experiments, the noise in the data used to fit $\hat{f}$ is independent of the noise in a test sample. It is worth noting that this would not be true if we were comparing with the same data used to fit the model, which explains the optimism of the training MSE.

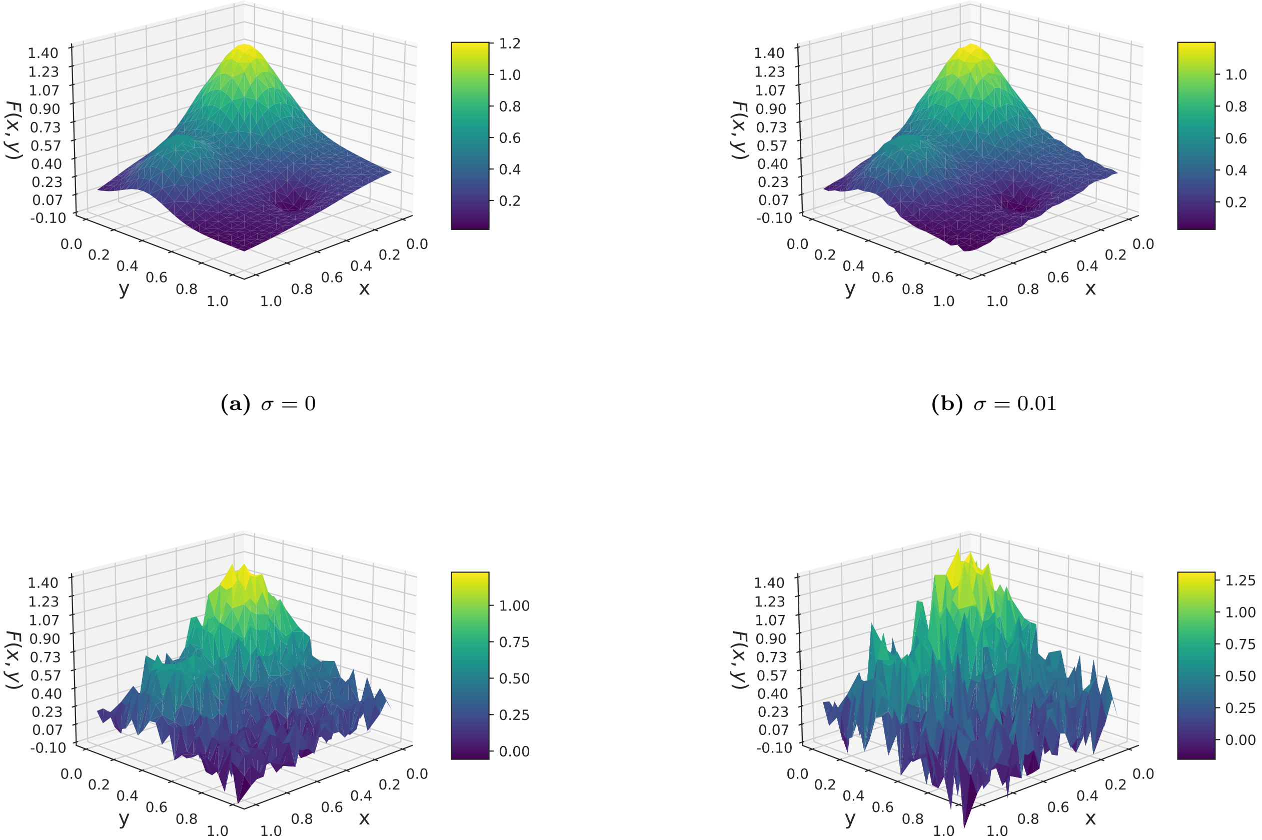
[3] In this appendix, we understand repeated indices to be summed over.

C Parameters for Artificial Data

To fairly compare between methods, a constant number of sampled points N from the Franke function (Eq. 17) and the noise magnitude σ needs to be set. We decided that $N = 600$ points should be sufficient to get enough data points in both the train and test set, while still being small enough to not bring in too much computational complexity. The points (x, y) are drawn from a uniform distribution $x, y \overset{d}{\sim} \mathcal{U}(0, 1)$ limited to their domain. Trough visual inspection, we decided on $\sigma = 0.1$ for the noise $\epsilon \overset{d}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. We found that this scale gave our function plenty of noise, while still being recognisable (Figure 19).

D Exploring Number of Bootstrap Rounds

Figure 20 shows that the distribution of MSE values derived from models trained on bootstrapped data set approaches a normal distribution as the number of bootstrap rounds increases, as the Central Limit Theorem states. In accordance with other results, one can see from the plots that when exceeding 400 rounds the distribution stabilises (see also Figure 6 for this result). As in Figure 20 the histograms show that the training data gives a smaller and more consistent MSE value than the testing data. This holds across number of bootstrap rounds.



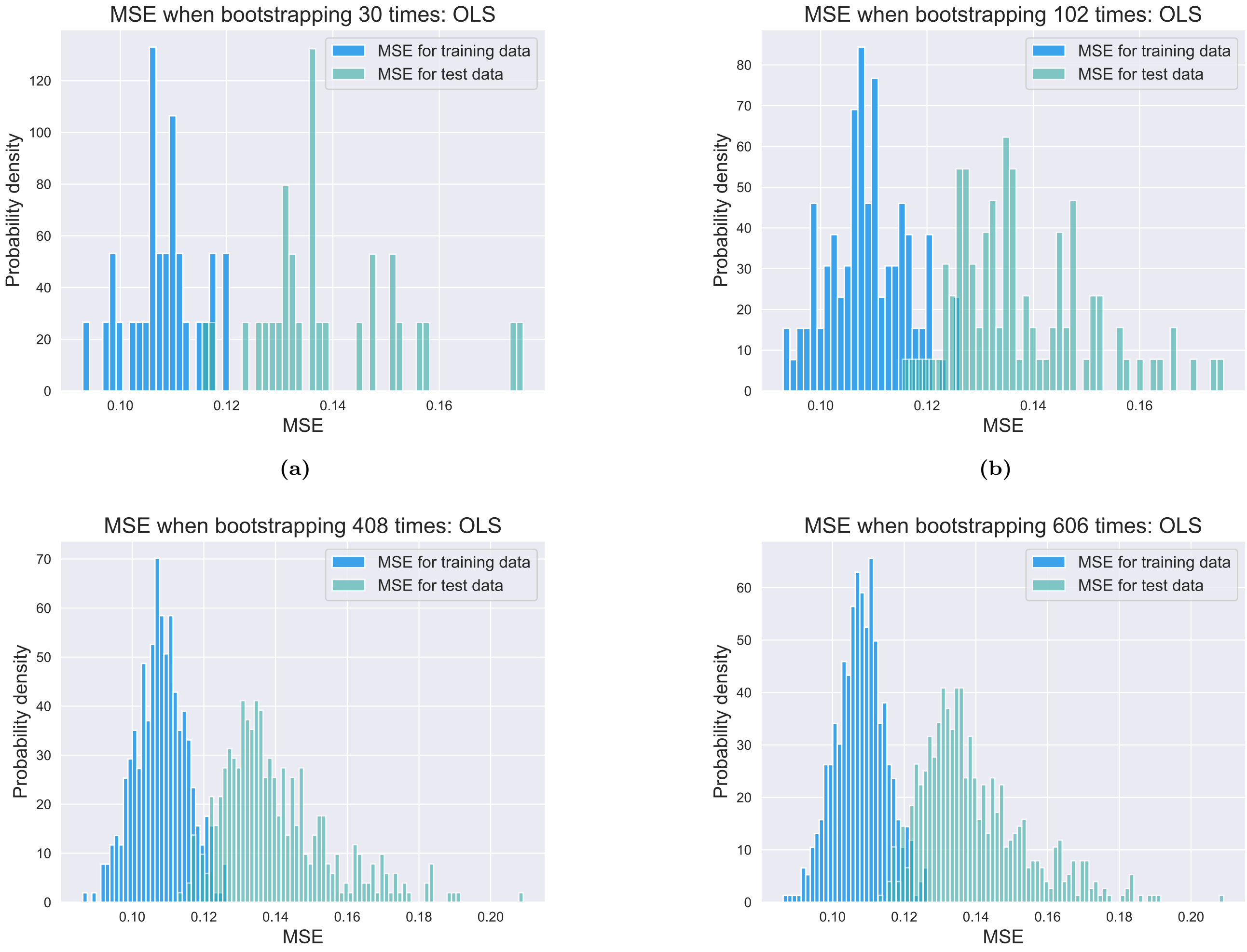
(a) $\sigma = 0$

(b) $\sigma = 0.01$

(c) $\sigma = 0.1$

(d) $\sigma = 0.2$

Figure 19: Showing Franke function with increasing noise, from a to d, $\sigma \in \{0, 0.01, 0.1, 0.2\}$. All figures plotted with $n = 625$ equidistant (x, y) points.



(a)

(b)

(c)

(d)

Figure 20: The histograms of MSE values for various number of bootstrap rounds.