Project 1 - Regression anaysis with resampling

Hanna Svennevik and Paulina Tedesco
github: https://github.com/hannasv/project1
fys-stk4155

September 2018

Abstract

In this article, we show that applying linear regression analysis to terrain data allows us to represent the main characteristics of the smoothed topography, although we are not able to describe all the features of the terrain with these simple models.

We fit the terrain data with polynomials of degree p, where $p \in [1, 5]$. As expected, an increase in the polynomial degree leads to a better fit and a better prediction. This is quantified with a lower MSE, and R^2-score closer to 1. However, the error depends strongly on the characteristics of the topography, for instance, the slope and the regularity of the terrain, whether there are large elevation differences in small areas.

On the other hand, an increase in the penalisation parameter for both Ridge and LASSO regressions does not improve the quality of the fit, demonstrating that the Simple Linear Regression, OLS, a particular case of these two types of regression with a regularisation parameter equal to zero, is the best model for fitting terrain data.

After comparing the error to the bias and the variance, we can affirm that the optimal polynomial degree is higher than 5. We are aware of that the main reason why we cannot reproduce all the characteristics of the elevation is that we use polynomials to predict a surface that does not have a polynomial behaviour. We suspect that the fit could be improved by using other functions, or methods that are non-linear. This would be implemented in a future work.

Introduction and motivation

Digital Elevation Models represents the Earth's topography, and these data are of major importance in all geosciences. The elevation of the surface controls a wide range of processes in the Earth System, such as the energy budget and biochemical cycles. As climatologist, we are interested in the fluxes of temperature, moisture, and air particles, also influenced by the topography. It is possible to use elevation data in coupled models to assess the impacts of climate change. Some recent studies in this field are: Gutiérrez et al. (2015)[12] ; Tolentino et al. (2017)[13] ; Arnold, D., Schicker, I. and Seibert, P. (2010)[14] ; Stuefer, M., Rott, H., Skvarca, P. (2007)[15] ; McGranahan, G., Balk, D. and Anderson, B. (2007)[16].

There are different sources of elevation data, ground surveys and cartographic surveys, airborne photogrammetric data capture and airborne laser scanning and Stereoscopic or radar-based satellite imagery. For this analysis, we use high-resolution data from the Shuttle Radar Topography Mission (SRTM), which is free and can be downloaded from the USGS EROS Data Center web page[7]. Despite the high quality of this data set, there are some voids, and it does not have a global coverage. Our goal is, therefore, to fill the gaps in the data using regression analysis. We propose three methods: Standard Linear Regression (OLS), Ridge, and the LASSO. With the exception of LASSO, these methods are linear and easy to implement. We will therefore write our own code which we will later compare with functions from the python library scikit-learn. An additional motivation is to completely understand these methods, by implementing and testing them.

In the first stage of this study, we fit polynomials to the two-dimensional Franke's function (see subsection Franke's function). We compare the three methods for different regularisation parameters and polynomial degrees, and select the best model. After having selected the most appropriate model, we assess its performance using the bootstrap as a resampling technique. A discussion of the bias-variance trade-off is also presented. The error is estimated with the MSE and the R-score metrics, which are calculated for all the possible models.

Thereafter, in a second stage, we repeat the same procedure with real topography data over a region close to Montevideo, Uruguay.

The report is organised with the following sections. Data, gives an overview of the data used in this report, the Franke's function and the real terrain data. Methods describes the principal methods used: the OLS, Ridge- and LASSO regressions, as well as the metrics and the bootstrap technique. Theoretical models are devoted to the regression analysis to the Franke's function; and Results to the analysis applied to real topography data of an image of Montevideo. Finally, the concluding remarks and future work are summarised in Conclusions.

Data

Self generated data and Franke's function

We begin this study by generating our own data set to assess the performance of the methods developed. The vectors x and y consists of 20 equally spaced points on the interval [0, 1]. Then, the Franke's function [3], a weighted sum of four exponentials given by equation (1), is applied to this vectors in order to generate a surface, z, which is going to be fitted using linear regression models. This surface is represented with contour lines in figure 1. The reason why the Franke's function is used in this paper is that it is widely used in geosciences to test various algorithms. Before running the regression analyses, an artificial noise is added to better simulate real data.

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

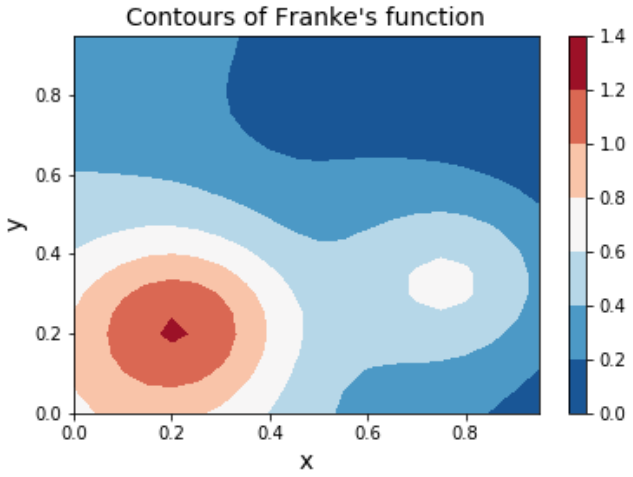

Figure 1: Contour lines for the Franke's function calculated applying equation (1) to self-generated data, x and y, in the range [0, 1]. The contour interval is equal to 0.2.

SRTM elevation data

After testing and comparing our functions to the ones developed for the scikit-learn library, we are ready to assess the models with real data, NASA's Shuttle Radar Topography Mission (SRTM)[6]. This topographic data set has the highest resolution, to be specific, 1 arc-second, or about 30 meters (98 feet). The SRTM digital elevation model (DEM) employed in this work is the void-filled version, known as "SRTM NASA Version 3", which uses lower resolution data to fill the gaps. The data is free for registered users and available for download from the USGS EROS Data Center[7].

The region chosen for the analysis is located over Montevideo (34531S, 561055W), Uruguay. The borders of the SRTM image downloaded from the USGS portal are represented with a green box in figure 2, and the red pin is positioned on the city of Montevideo. The image itself is shown in figure 3, with dark grey tones for the lowest elevations, like the the Atlantic Ocean and the rivers, and lighter grey tones for higher elevations. The image covers a larger region than the department of Montevideo, some areas of Canelones, to the north, and San José to the west of the river Santa Lucía. Montevideo is a relatively small coastal city, situated on the north shore of the Río de la Plata, which behaves as an estuary where fresh water from the river mixes with the sea. It is also a flat city, with an average elevation of 43 m[8], and its architecture consists mostly of low buildings. The Bay of Montevideo forms a natural harbour, and it is located close to it is the highest point of the city (134 m), the Fortaleza del Cerro.

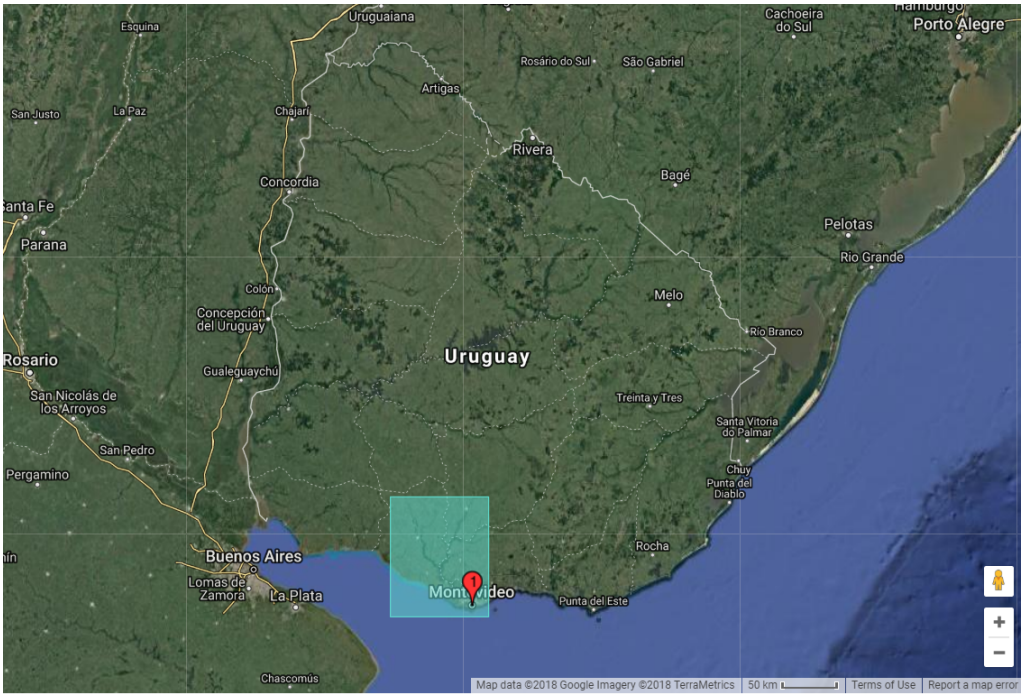

Figure 2: Satellite image of Montevideo and limits of image selected. Google Maps downloaded from the USGS EROS Data[7].

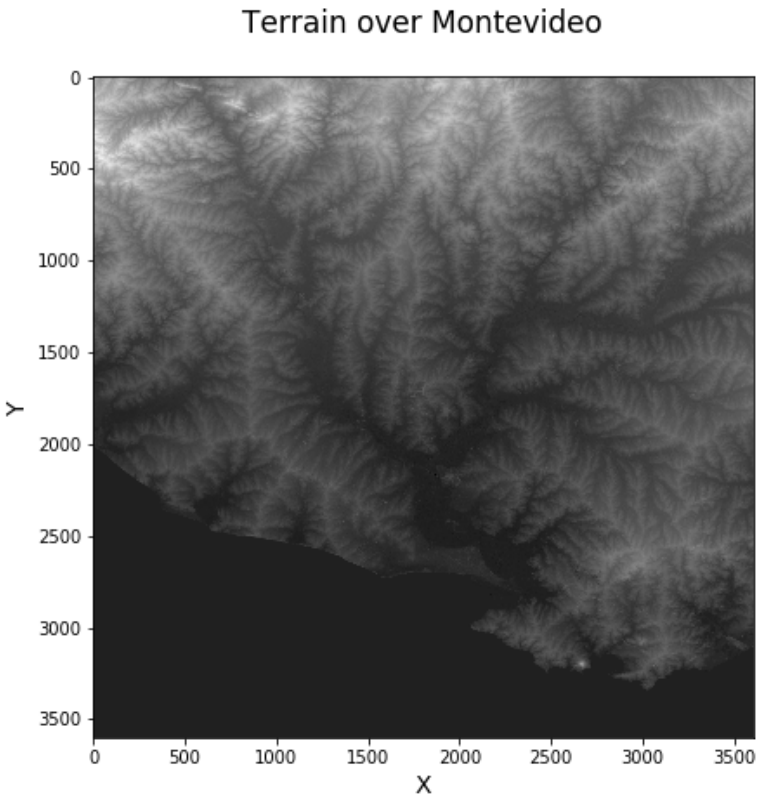

Figure 3: SRTM image of Montevideo, Uruguay, and surroundings. The original .tif file was processed in Python and saved as a .png file that is shown in this figure. Source: USGS[7].

Methods

Regression analysis allows us to find the relationship between a response and one or more explanatory variables. In this article, we will investigate the relation between the height of the surface, z, based on the coordinates x and y. We will evaluate the performance of two different types of linear regression, OLS and Ridge, and one case of non-linear regression, the LASSO. The perks of linear regression are that there exists an analytic expression for the minimum of the cost function, making these methods simpler to implement, and lowering the computational cost since we don't have to calculate any derivatives numerically. These methods are well established and discussed in the literature, and we will therefore only give a short description in this section. For more details, see the bibliography cited in this article.

[10]

Ordinary least squares

$\hat{Y}$ denotes the predicted model given the input $\mathbf{X}$. $\hat{\beta}_0$ is the intercept also known as the bias in machine learning. The best fit is given by the $\hat{\beta}$-values that minimise the residual sum of squares, which is the cost function in this case. The minimum is found by differentiating the RSS with regards to β and setting it equal to zero. Finally, solving for $\hat{\beta}$ results in equation (4). This has a unique solution when $\left(\mathbf{X}^T\mathbf{X}\right)$ is non-singular. A matrix is non-singular when X^TX has full rank, then all the columns are linearly independent and the matrix is invertible. [1]

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{2}$$

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

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y} \tag{4}$$

Ridge regression

When the amount of predictors is large relative to the number of observations, a penalisation term is often added to penalise those features that add little to the prediction, by adding a penalty term, λ, to the diagonal of the matrix. The ridge regression imposes a size constraint on the coefficients and reduces the number of correlated coefficients which could result in a singular matrix. [1]

$$\hat{\beta}^{ridge} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y} \tag{5}$$

The LASSO regression

The difference between the LASSO and ridge regression is that the regularisation term in the cost function is in absolute value. The $\sum_{j=0}^{p} \beta_j^2$ is replaced with $\sum_{j=0}^{p} |\beta_j|$. Lasso is non-linear and there is no closed for expression for $\hat{\beta}$ in this case. [1] The expression for Lasso on Lagrangian form is:

$$\hat{\beta}^{lasso} = \underset{\beta}{argmin} \left\{ \frac{1}{2} \sum_{i=1} \left(y_i + \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right\} \tag{6}$$

One substantial difference between ridge and the LASSO is that the former shrinks the coefficients, but not to zero, whereas the latter does not only penalise the coefficients but actually set them to zero if they are irrelevant.

Performance metrics

In order to quantify the performance of a model we use metrics. In this article, we use both the mean square error, MSE, and the R^2-score. MSE is the average squared distance between the predicted and the true value. The R^2-score is a measure on how well future samples are likely to be predicted by the model. They both result in a good score when the models performance approaches the true value. Then the MSE goes to zero and the R^2 goes to one. The metrics can be calculated by the following equations. [10]

$$MSE(\hat{y}, \hat{\tilde{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 \tag{7}$$

$$R^2(\hat{y}, \hat{\tilde{y}}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2} \tag{8}$$

where mean value of $\hat{y}$ is defined as $\bar{y} = \frac{1}{n} \sum_{i=0}^{n-1} y_i$.

Design matrix

In order to fit a polynomial of degree p to the data, we need to expand the basis to include polynomial representations [1]. The general form of a design matrix, $\mathbf{X}$ when fitting a polynomial of degree p to a function of

two variables x and y are given by equation (9).

$$\mathbf{X} = \begin{pmatrix} 1 & \mathrm{x}_0 & \mathrm{y}_0 & \mathrm{x}_0^2 & \mathrm{x}_0\mathrm{y}_0 & \mathrm{y}_0^2 & \cdots & \mathrm{x}_0^p & \mathrm{x}_0^{p-1}\mathrm{y}_0 & \cdots & x_0 y_0^{p-1} & \mathrm{y}_0^p \\ 1 & \mathrm{x}_1 & \mathrm{y}_1 & \mathrm{x}_1^2 & \mathrm{x}_1\mathrm{y}_1 & \mathrm{y}_1^2 & \cdots & \mathrm{x}_0^p & \mathrm{x}_0^{p-1}\mathrm{y}_0 & \cdots & x_1 y_1^{p-1} & \mathrm{y}_0^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 1 & \mathrm{x}_n & \mathrm{y}_n & \mathrm{x}_n^2 & x_n y_n & \mathrm{y}_n^2 & \cdots & \mathrm{x}_n^p & \mathrm{x}_n^{p-1}\mathrm{y}_n & \cdots & x_n y_n^{p-1} & \mathrm{y}_n^p \end{pmatrix} \tag{9}$$

The number of columns, T, in the design matrix for a given polynomial degree, p, is determined by equation (10).

$$T = \sum_{i=1}^{p+1} i = 1 + 2 + 3 + ... + p + (p+1) = \frac{(p+1)\,(p+2)}{2} = \frac{p^2 + 3p + 2}{2} \tag{10}$$

Bias-Variance trade-off

When using machine learning to make predictions, we deal with finding a balance between modelling the training data very closely (trying to capture most of the features) and fitting it closely (over-interpreting outliers and irregularities that could lead to poor predictions of new, unknown data) [9]. Shortly, we want to avoid overfitting, this is, train the predictive model approximating the training data too closely; but we also want, in the other extreme, to avoid underfitting ignoring important features of our data, when choosing a too simple model. This problem is known as the bias-variance dilemma.

On the one hand, we have the bias, which refers to the underfitting, and measures the deviation of the expectation value of the estimator. On the other hand, we have the variance, which refers to the overfitting, and measures how much the estimator fluctuates due to the finite-sample effect. When trying to fit a data set with a model that is not complex enough, the predictor will not be able to capture all the features, leading to a high bias. This does not mean that the fit is wrong, it is simply the best performance of a simple model. On the other extreme, trying to fit the data with a model that is too complex will lead to high variance and, thus, low variance since it adapts too much to the data. Again, it is not the training that is wrong, but the selection of the model with its inherent complexity. Consequently, there is a trade-off between the bias and the variance. This means that we have to sacrifice one to lower the other since both depend on the complexity of the model.

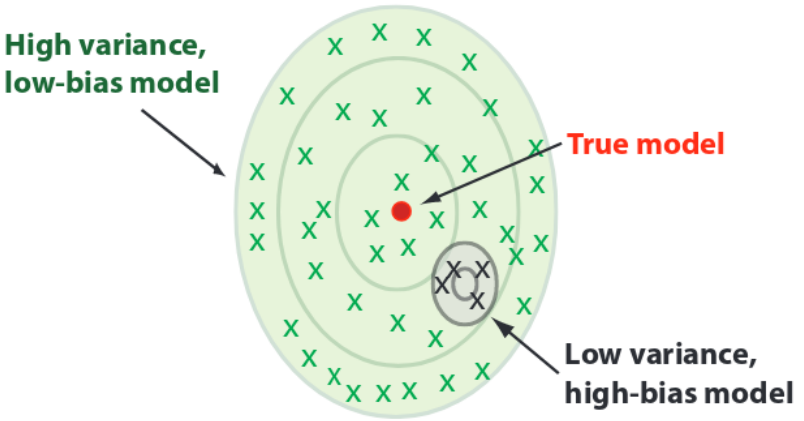

Figure 4: Illustration of the bias-variance trade-off. Source: Mehta et al. (2018)[4].

Figure 4[4], illustrates the bias and the variance of two hypothetical models, where the red dot is the true model. The model represented with green crosses correspond to a model with high variance (random error) since all the points are dispersed around the real model, but with a low bias because they are centred on the true

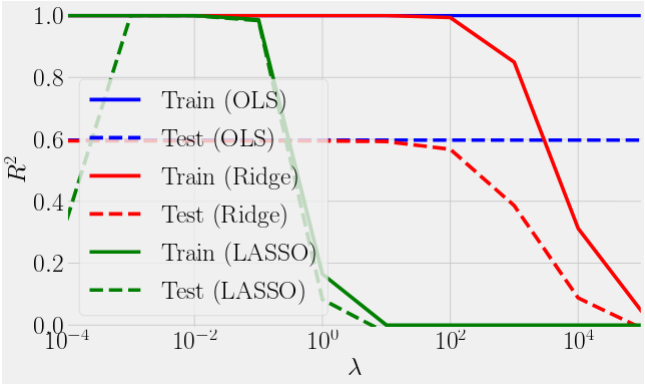

Figure 5: Example of performance of OLS, Ridge and LASSO regressionon as measured by the R2coefficient. Source: Mehta et al. (2018)[4].

model. The model represented with black crosses have a high bias (systematic error), since their centre does not coincide with the true model, but the distance between the points is small and therefore it has a low variance.

The test and the training errors indicate whether there is an under- or overfitting (see figure 5). As explained before, because low complexity models lack the ability to describe well the data, they result in poor accuracy, and consequently high training and test errors. Contrarily, high complexity models, which adjust too much to the training data, but fails generalising the test data, resulting in low training error and high test error. In conclusion, the best model can be found where the test error reaches a minimum.

Real data is also affected by random fluctuations called noise (ϵ), with unknown origins such as human errors or measurement errors. So, we assume that the measured data y is the sum of the theoretical perfect function ($f(x)$) than we want to model and the random error ϵ: $y = f(x) + \epsilon$. For constructing an estimation of the function, f, we need to ignore the unknown noise. How much we ignore is not trivial, cause we do not want to ignore real data, thus, the bias-variance dilemma.

The error can be decomposed in three terms[1], the bias term, the variance term, and the random noise (see figure 6).

$$error = bias + variance + \epsilon \tag{11}$$

Measuring the error is hard when the data set is small. Therefore, we use the bootstrap as a resampling technique (see subsection Resampling).

The problem of high bias can be solved by selecting a more complex model. In the opposite case, a way to deal with high variance is to add a regularisation parameter that penalises for the increase in complexity (see subsections for ridge and LASSO regressions). In theory, an infinite amount of training data will result in a low bias model. This is usually not the case when working with real data. When the sample is limited, the arising from the finite-sample (e.g. variance) may cause a simple model to outperform a complex one. [4]

Resampling and splitting

Resampling is used to assess the performance of a model. The resampling technique used in this article is the bootstrap, which consists of extracting observations from a given sample with replacement, generating new sets with the same size as the original sample, and repeat this N times. Efron (1979)[2] prove that in the limit of large samples, the bootstrap method provides an estimation of the true variance. This resampling methods can be computationally expensive, but they have evolved with the advent of cheap and powerful computing. In this work, it was implemented by randomly extracting with replacement rows of the original design matrix (X), generating a *new* a new matrix with the same size, ($\tilde{X}$), and then repeating this 100 times. Training the model for several data sets reduces the bias considerably. The data is first divided into training and test data, and then the bootstrap is performed on the the training set.

Decreasing the size of the test set may result in a substantial increase in the variance. The statistics depends

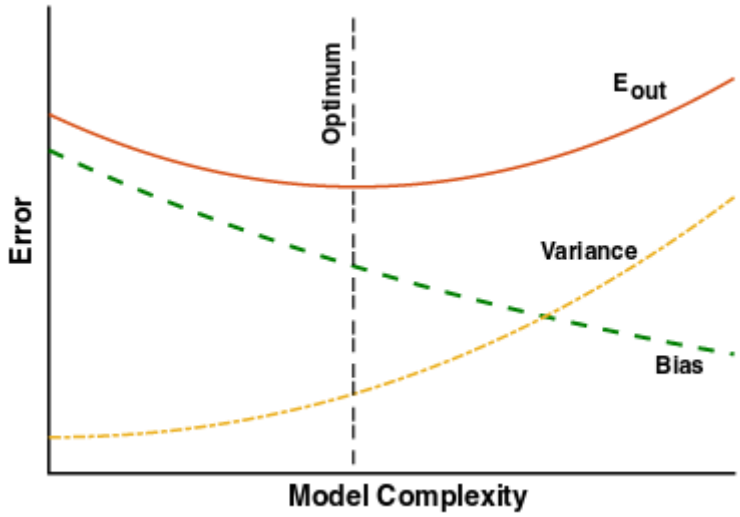

Figure 6: "Bias-Variance trade-off and model complexity. This schematic shows the typical out-of-sample error E_{out} as function of the model complexity for a training dataset of fixed size. Notice how the bias always decreases with model complexity, but the variance, i.e. fluctuation in performance due to finite size sampling effects, increases with model complexity. Thus, optimal performance is achieved at intermediate levels of model complexity." Source: Mehta et al. (2018)[4].

on the instances in the datasets. Since the data changes for each resamle it follows that the statistics gets altered. When we discuss the general performance we discuss the average statistics of all the resamplings. Often refereed to for all the boots since we are using the bootstrapping technique.

Let N denote the number of boots used in a resample. Then we have the followig expression for the bias.

$$Bias = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{N}\sum_{j=1}^{N}\tilde{y_{j,i}}\right) \tag{12}$$

Confidence intervals

In this section, we present the methodology followed to calculate the 95% confidence intervals.[5] If we assume that our sample is drawn from a normal distribution with mean μ and variance σ^2, both unknown, we can use the sample mean to estimate

$$\mu$$

, this is:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{13}$$

and the variance can be estimated as:

$$Var = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{14}$$

The standard error (SE) can be computed as

$$SE = \frac{\sqrt{VAR}}{\sqrt{n}} \tag{15}$$

and we can use the following expression to estimate the 95% confidence intervals.

$$\bar{x} \pm t_{96} SE \tag{16}$$

where $t_{96} = 1.984$

Structure and implementation

All the material used in this article is available at https://github.com/hannasv/project1. Each regression method implemented as a class in the file algorithms.py, with their own class functions fit and predict. With the exception of LASSO regression, all regression methods and functions are implemented from scratch. The script utils.py contains functions accessed from the notebooks and code used in this report. All the functions in utils.py where tested in the file *test_project1.py*. We have structured the solution into two parts. The first part consists of finding the optimal hyperparameter λ. This is done with a call to the *model_comparison0* function, which creates one instance of a gridsearch object for each regression method given as an input. The gridsearch object fits all of the methods and returns itself. The gridsearch object now contains all the properties of the regression. These can now easily be accessed in *model_comparison* which returns a dictionary of the performance metrics. This function is illustrated in a box below.

```
# Experimental set-up.
models = {
    "ols": algorithms.OLS,
    'ridge': algorithms.Ridge,
    "lasso": algorithms.Lasso
}
param_grid = {
    'ols': [0],
    'ridge': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
    'lasso': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0]
}
nboots = 100
```

```
def model_comparison0(models, param_grid, X, z, split_size=0.2, verbose=True):

    for each model:
        Create a gridsearch object containing the arrays on lmd-values.
        Fit the gridserach object. Now this contains mse, r2, coeffesients of the fit

    finds the best mse and return the coefficients of this fit.
    results = {"mse_test": mse_test, "mse_train": mse_train, "r2_test": r2_test, "r2_train":
                                                    r2_train}

    return results, z_pred_best
```

The second part of the problem which include resampling and calculations of average mse, r2, model variance, bias and confidence interval of the model. These calculations can be performed by a call to the function *resample*(). The structure of the function is illustrated below. All the results can be found in Test_functions.ipynb.

```
def resample(models, lmd, X, z, nboots, split_size=0.2):

    for each boot:
        predict z based on test and train data

    Compute bias, var, beta, mse_test, mse_train, ci_beta

    return z_test, z_pred_test, bias, var, beta, mse_test, mse_train, ci_beta
```

In addition to these we have made two scripts. One for utilities and another script which tests all the units in the utils.py. In order to execute the tests you make the following call in your command line. The tests are made in order to make the program more reliable and for us to be able to test units of the code before its finished.

The reader can find some calculations that demonstrate the code in the notebook Test_functions.ipynb.

```
pytest test_project1.py
```

Theoretical models

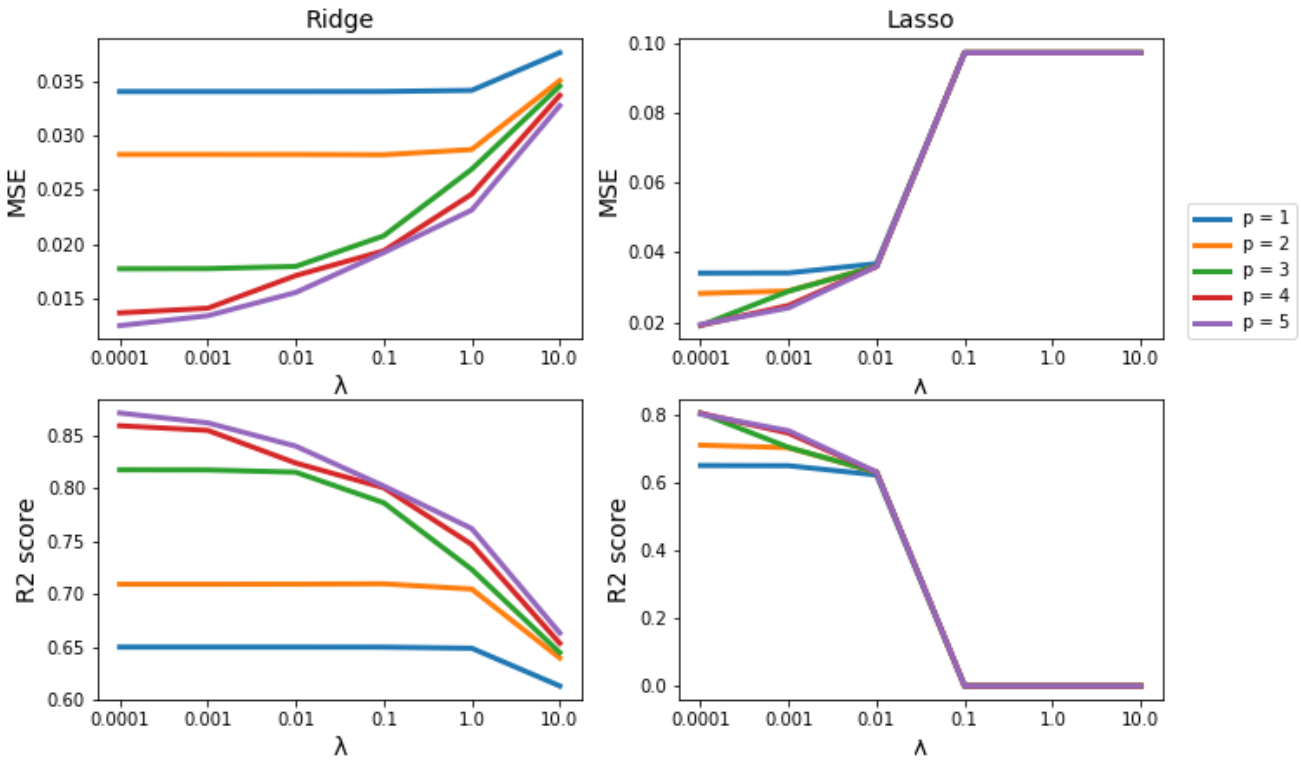

Figure 7: Illustration of the performance metrics (MSE and R^2) on the Franke's function for several λ and $p = 1, 2, 3, 4, 5$.. The methods analysed are ridge and the LASSO

Our first goal is to determine the best model that fits our data by comparing the three approaches discussed above: OLS, ridge, and lasso regression. We start by fitting the Franke's function applied to equally spaced data points. Then we compute the error for each model. Figure 7 shows the MSE and R^2-scores for the ridge regression and the LASSO as function of λ, here represented with a logarithmic axis. Each curve, corresponding to a different polynomial order ($p \in [1, 5]$), is represented with different colours in the same panel. The training set, which consists of 80% of the data, is used to fit the models, and the remaining 20% is left for the prediction and computation of the scores. Even though the data is split randomly, and figure 7 shows only one example, the leading results can be generalised. Similar figures for up to 1000 can be found in the Appendix.

From the analysis of figure 7, we can conclude for the ridge- and lasso regression that the scores are improved when λ tends to zero, suggesting that the best model for these data is the OLS. This means that adding a penalisation term, i.e., shrinking the regression coefficients, does not decrease the error in this particular case. Among the cases which are plotted in fig. X the best model is the case with highest degree polynomial and lowest regularisation parameter. Hence, we select the the simplest of the three approaches discussed in this paper.In the appendix you can find a similar figure for higher values of regularisation parameter.

We also observe in Figure 7 that the curves for the different polynomial degrees get closer to each other when λ increases. In the case of the LASSO, the curves seem to converge more rapidly than in the case of the ridge regression. There are no appreciable differences in MSE nor R-squared for values greater than 0.01 for the LASSO, but for the ridge regression, this small differences in errors are achieved for greater values of λ (not shown in this figure). Also, in the case of LASSO, two plateaus can be identified, one for $\lambda < 0.01$ and the other for $\lambda > 0.1$, separated by a region of rapid increase in error; whereas in the case of ridge, then we can appreciate an almost monotonically increase of the error in terms of $\lambda > 0.1$. The latter applies for higher polynomial degrees. Notice that there is less variation with lambda for $p = 1$ than for $p = 2$, and so on; in fact, when splitting the data randomly, one can find cases where $MSE(\lambda = 0.0001, p = 1, 2) > MSE(\lambda = 0.1, p = 1, 2)$ Another interesting characteristic is the fact that the curves corresponding to higher values of p present a greater slope for the lowest range of penalisation parameters.

The MSE as function of p for the OLS approach is plotted in figure 8. A decrease with p is observed in

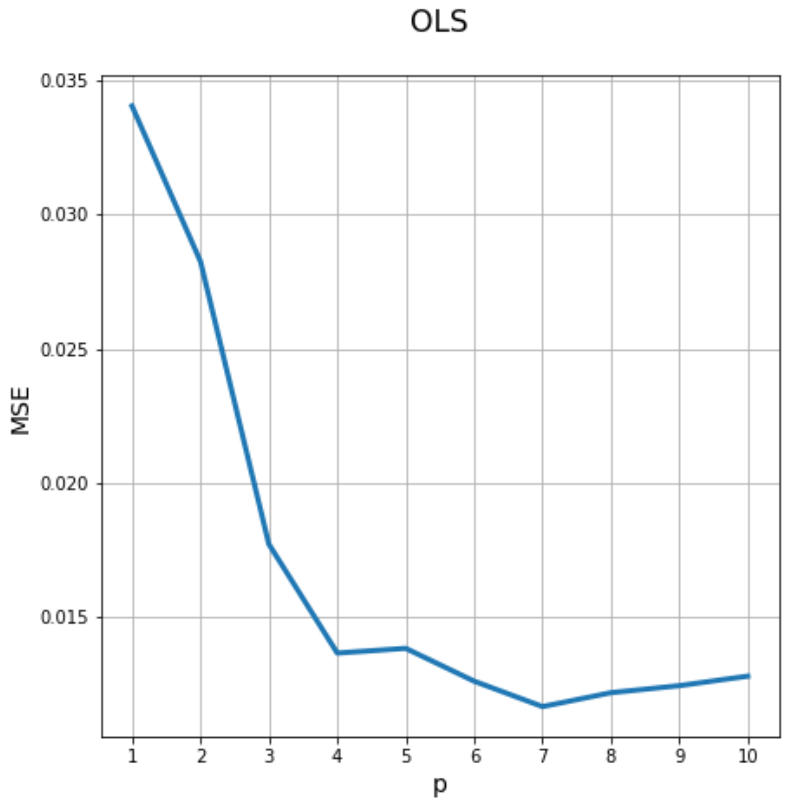

Figure 8: MSE score as a function of p for the OLS approach applied to the Franke's function.

this figures, as expected, but, this function has a peak at $p = 17$, reaching MSE levels higher than 4000 (not shown in this figure). In the range $p \in [1, 5]$ that we study in this work, the MSE decreases monotonically.

Mehta et al. (2018) affirm that it is a good practice to plot the performance as function of since the regularisation parameter affects the ridge and the LASSO regressions (see example in figure 5). The test and training errors of the three schemes, OLS, ridge, and LASSO with $p = 5$ are represented in figure 9; the left panel shows the MSE and the right panel shows the R^2-score. Along the lines of the previous results, this figures reaffirm that the OLS presents the lowest errors and is the model that best fits and predicts the Franke's function; for the other two schemes, the error increases with . Both panels indicate that the LASSO error achieves a plateau and stops increasing when $> 0, 1$, whereas the ridge error increases and its curve becomes steeper for higher values of . In general, the training error is expected to be higher than the test error, since, in the case of the training set, z is being predicted for the same data points that were used to train the model. However, we might find some exceptions as in this figure. On the one hand, the left panel shows that the MSE for the test set is lower than the one for the train set for < 1.0 in the case of ridge regression, and the same observation can be made for < 0.001 in the case of the LASSO; on the other hand, the right panel shows higher values of R-squared on the test set for all the values of in the case of ridge regression, and for < 0.01 in the case of the LASSO. There might be several explanations for this behaviour. Although not proved here, we think that some reasons why the test error is lower than the training error could be that the test sample (20% of the data) is smaller than the training sample, and as consequence has less noise; it can also be possible that the penalisation parameter, in some intervals, removes overfitting (causing a higher error for the training set) and that the models generalise well (leading to low test errors). We do not think it can be the random split of the data because we repeated the process (not shown) obtaining the same results.

A scatter plot of the generated data and the prediction using OLS and $p = 5$ is represented in figure 10. It is not easy to see whether there is an over- or underestimation by just looking at the pictures. In general, the points are centred around the 1:1 line. It seems like the variance does not vary significantly with z. The point in the plot with the highest value of original z looks suspicious, and it would be interesting to determine if it is an outlier with formal calculations. It also looks like there is a slightly higher density of points on the left side

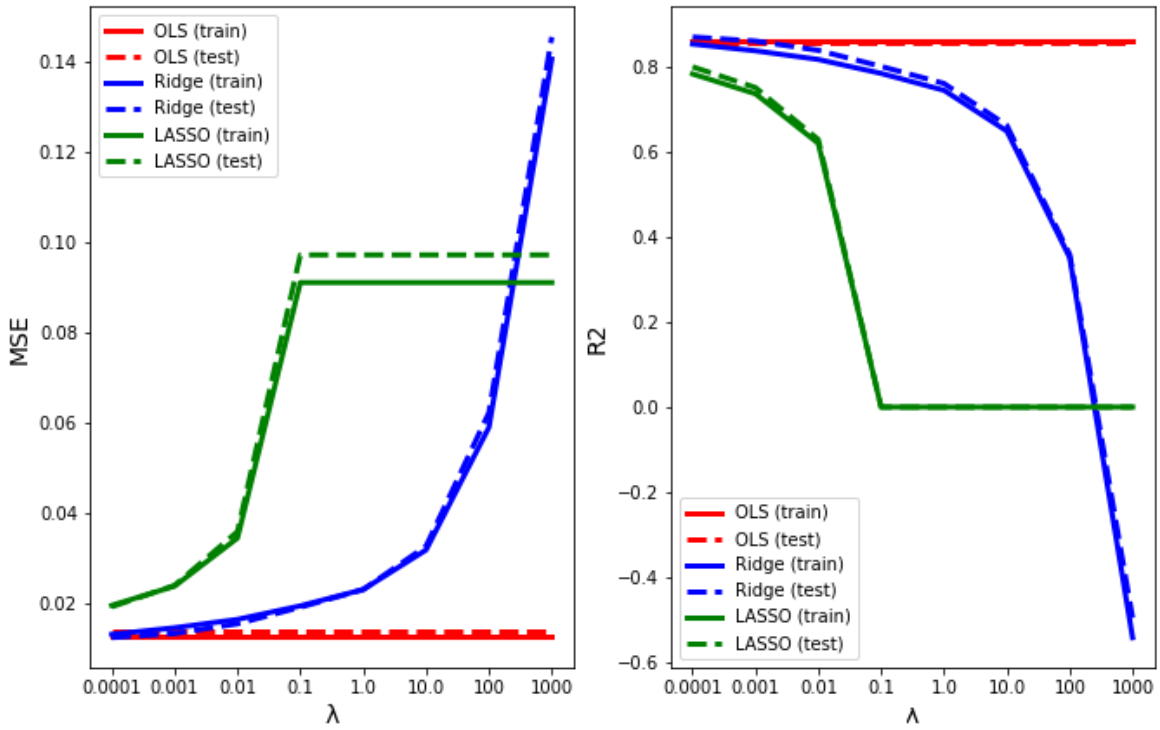

Figure 9: Performance of the OLS, Ridge and LASSO regressions on the Franke's function for different penalisation parameters (logarithmic axis) and $p = 5$. This figure was obtaining by resampling the data set 100 times. For more details, see notebook file.

of the plot (where the points with lower z are located) than on the right side.

In this section, we compared and discussed the performance of the OLS, ridge, and the Lasso regression, varying the penalisation parameter and the polynomial order of the functions of the design matrix. All the results presented above indicate that the best model for fitting and predicting the data generated with the Franke's function is the OLS with a polynomial degree equal to 5. Consequently, we study the bias-variance tradeoff for this model and show the confidence intervals of the regression coefficients in table 1. The 95% confidence intervals are calculated using equation the student's distribution as in equation (15).

Figure 11 shows the test error, estimated by the MSE, the bias and the variance of the OLS as function of the polynomial degree (complexity of the model) up to $p = 10$. Recall from the discussion in the bias-variance section, that our aim is to determine the optimal complexity. In principle, we limited this study to the range $p \in [1, 5]$, obtaining a minimum bias and variance for $p = 4$ by looking at the error. For curiosity, we also plotted the for higher polynomial degrees. The variance is relatively small and steady for $p \leq 9$, and then increases rapidly affecting the MSE, while the MSE and the bias have a similar behaviour, decreasing considerably from $p = 1$, presenting some irregularities form $p = 4$ to $p = 9$, and then increasing to $p = 10$. The bias and the MSE curves present three local minimums at $p = 4, 7, 9$, and the curves apart from each other as a consequence of a small increase in the variance. A figure for higher polynomial orders can be found in the Appendix. This indicates shows the enormous increase in the error for $p \geq 10$.

We can also guarantee that the MSE, the error, is smaller than the sum of the bias and the variance. These calculations are done in this notebook, $\text{Project1}_F YSSTK4155.ipynb$. Even though, the difference is much smaller than the values of the errors, meaning that the random noise is very almost insignificant.

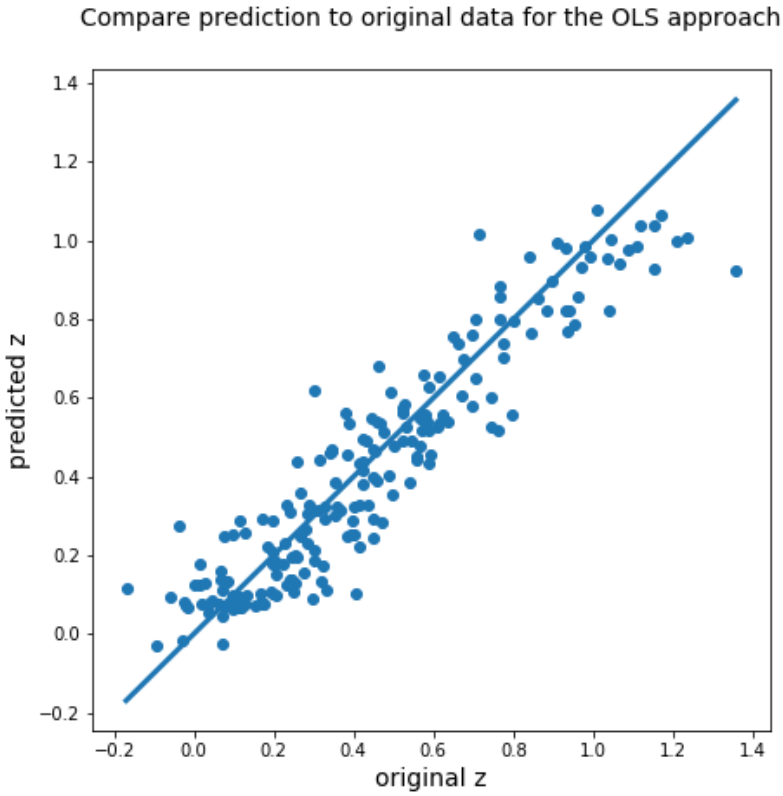

Figure 10: Scatter plot of the predicted test data and the original test points for the Franke's function and 100 resamplings. The model used is the OLS with $p = 5$.

Terrain	**lower ci**	**upper ci**
β_0	$1.0453 \cdot 10^1$	$1.4614 \cdot 10^1$
β_1	-1.2180	$2.7877 \cdot 10^{-1}$
β_2	$2.9842 \cdot 10^1$	$3.1260 \cdot 10^1$
β_3	4.0149	$1.1848 \cdot 10^1$
β_4	$-8.5385 \cdot 10^1$	$-7.9711 \cdot 10^1$
β_5	$-1.7606 \cdot 10^2$	$-1.6840 \cdot 10^2$
β_6	$5.9238 \cdot 10^1$	$7.7307 \cdot 10^1$
β_7	$-3.5879 \cdot 10^1$	$-2.3306 \cdot 10^1$
β_8	$3.3064 \cdot 10^2$	$3.4428 \cdot 10^2$
β_9	$3.4802 \cdot 10^2$	$3.6668 \cdot 10^2$
β_{10}	$-1.8962 \cdot 10^2$	$-1.7080 \cdot 10^2$
β_{11}	$2.2892 \cdot 10^2$	$2.4284 \cdot 10^2$
β_{12}	$-2.1445 \cdot 10^2$	$-2.0092 \cdot 10^2$
β_{13}	$-3.3010 \cdot 10^2$	$-3.1451 \cdot 10^2$
β_{14}	$-3.3757 \cdot 10^2$	$-3.1684 \cdot 10^2$
β_{15}	$1.0293 \cdot 10^2$	$1.1022 \cdot 10^2$
β_{16}	$-1.4649 \cdot 10^2$	$-1.4026 \cdot 10^2$
β_{17}	-4.4288	1.9696
β_{18}	$1.4694 \cdot 10^2$	$1.5338 \cdot 10^2$
β_{19}	$6.6876 \cdot 10^1$	$7.3893 \cdot 10^1$
β_{20}	$1.1305 \cdot 10^2$	$1.2150 \cdot 10^2$

Table 1: 95% confidence intervals for OLS with $p = 5$ based on real data.

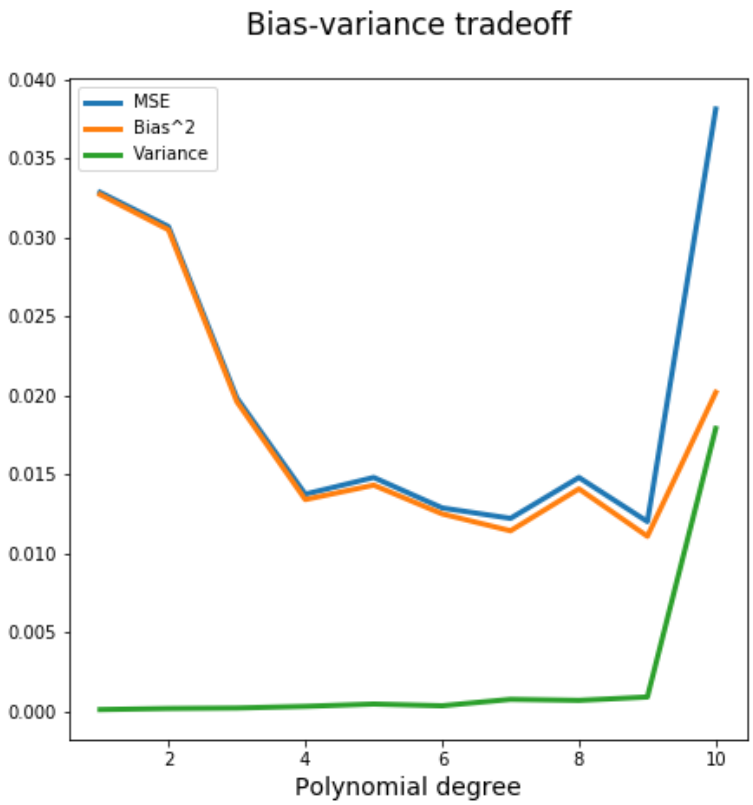

Figure 11: Bias-variance trade-off of the OLS applied to the Franke's function. MSE, bias and variance are plotted as function of the polynomial degree. 100 resamplings were performed to generate this figure.

Results

All the material used in this article is available at https://github.com/hannasv/project1. Github.

Real data

The image of Montevideo is too large to be processed in python with our own codes in our machines. Hence, some random patches, of size 90x90 points, were selected from the image. One of the patches is represented in figure 10, and the rest can be found in the Appendix. The real terrain surface and the fitted terrain data are depicted in the right and the left panels respectively. In order to generate this figure, the data is not split in training and test sets, thus the training of the model and the prediction are performed on the same data points. Figure 10, and the analogous figures in the appendix, show that the implementation of the OLS using polynomials of fifth order allows us to reproduce a smoother surface than the real terrain data, missing some small-scale variations. Furthermore, in almost all the examples, the values of the extremes are lower, in absolute value, in the predicted version compared to the original. Whether this is a good approximation, should be discussed in terms of the application. However, it is important to mention that this could be improved by using higher orders of polynomials or other functions to fit the data, as well as other non-linear methods.

The metrics of the respective patches are shown in table 2. Notice that, in the case of a flat terrain (over the sea), our model gives a perfect fit. In general, the MSE is poor, but the R2-score is higher than 0.76 in for all the examples. Determining the best size of the sample is not trivial in this case. The same experiment was executed for patches of size 50x50. In all cases, the R2-score, and, especially, the variance and the bias were improved considerably; nevertheless, the MSE was only lower for two of the five batches. This may be explained by the height variations in some regions. In a larger sample, if the terrain is irregular, it is more likely to increase the amplitude, in other words, the difference between the maximum and the minimum level increases, as the probability of the presence of high slopes will increase. In this areas, a polynomial will fit the terrain worse than where the surface is smooth. We believe that this should be analysed carefully. Unfortunately, because of memory issues, we are not able to run our codes with more data points.

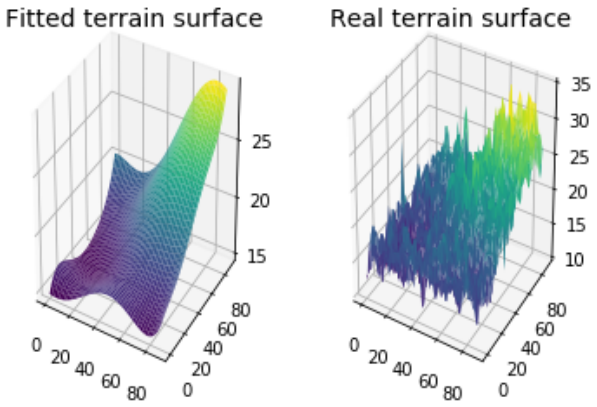

Figure 12: Fitted and real terrain surfaces of random patch of the image of Montevideo. The model used is the OLS for $p = 5$. This will be referred to as patch number 4.

Regression model	**data**	**mse**	**r2**	**var**	**bias**
Terrain	patch 0	8.4708	0.7622	27.1439	35.6147
	patch 1	0	nan	0	0
	patch 2	10.5735	0.8225	48.9812	59.5547
	patch 3	4.5474	0.8537	26.5308	31.0782
	patch 4	3.1690	0.8059	13.1554	16.3243

Table 2: The average metrics after a hundred resamples with $p = 5$.

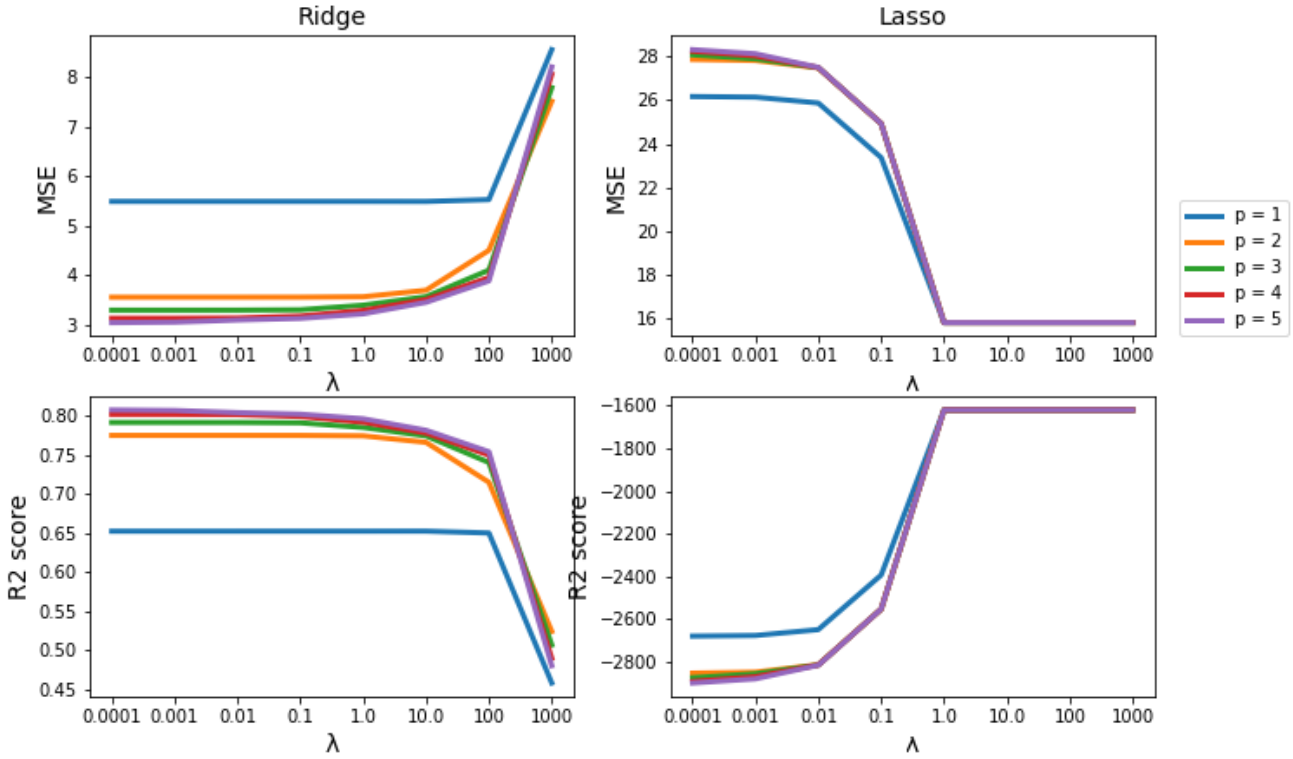

Figure 13: MSE score as a function of p for the OLS approach applied to the SRTM image, patch 4.

Many of the results of the following part of the analysis will be done for the patch 4 in figure 12 for computational reasons and to avoid an excessively long report. Meanwhile, when possible, we also compare the different patches. Figure 13 is analogous to figure 7. For all the polynomial degrees, in the case of the ridge (the LASSO) regression, the error is lower for lower (higher) regularisation parameters. Another interesting result is that, in the case of the LASSO, a fit with a one order polynomial leads to a lower MSE and a higher R2-score. This means that this simpler model reduces the error, even though the variance is high. From this figure, we deduce that the OLS would be a better approach than the ridge regressions since it suggests that putting $= 0$, this is, not using any penalisation, would give better results. However, we still need to compare the OLS with the LASSO, although, after looking at the figure and comparing ridge with the LASSO, we suspect that the scores for the OLS will be better than the LASSO scores.

Figure 14 shows a considerable variation of MSE for the different patches. The violet line, patch 4 is the one used in the previous two figures and, as suspected, the error is smaller than with the LASSO approach. The orange line corresponds evidently to the patch over the sea. Patches number 0 and 2 are located over more irregular areas, where a polynomial will not represent well the features of the terrain, compared to patches 3 and 4, where the small-scale changes are smaller despite of the slope (see appendix). We cannot find any clear evidence of an existing relationship between the average height of the patches and the performance, although we believe that the noise in the data in some mountain areas where, for instance, shadows can affect the measures of the height, might lead to poorer fits. This could also be the case of the voids filled with lower resolution data, or areas with low reflectivity such as deserts can provoke poor observations.

The performance of the three approaches is exhibited in figure 15, for $p = 5$ and patch number 4. We see clearly that the LASSO regression performs worse than the OLS and the ridge. Because of the scale, it is hard to distinguish in the right panel the OLS from the ridge lines, but for the test sample, these can be inferred from the previous figures, or by looking at figure 23 in the appendix that shows the MSE test and train scores only for OLS and Ridge. These curves get very close together for small values of λ, but for high values the error of the ridge approach increases while the OLS error is constant. In both cases, the training error is lower than the test error (see discussion above for the Franke's function). In this case, the relative sizes of the training and test sets do explain the lower test errors. Although not shown here, this was tested for different sizes (see the

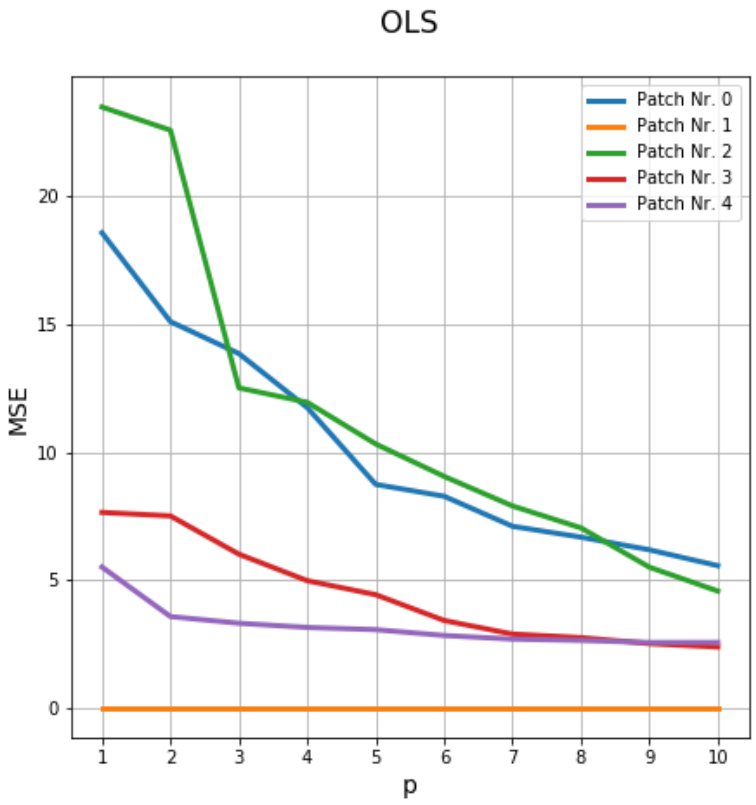

Figure 14: MSE score as a function of p for the OLS approach applied to the patches of the SRTM image. The different lines represent the random patches selected from the image. Patch number 4 is the selected for figure number 10.

jupyter notebook in GitHub). The regression coefficients of the OLS model for $p = 5$ are shown in table 3, as well as their confidence intervals calculated using equation (15), resampling the data 100 times.

Figure 16 is a scatter plot of the predicted and the original data. We can not distinguish any particular pattern or outliers. All the points are distributed around the 1:1 line.

The bias-variance trade-off for the OLS approach, $p = 5$, and patch 4 is represented in figure 17. Again, it is important to clarify that we are only exploring models with complexity up to a fifth polynomial order. From these results, we would select a model with polynomial order 4 or 5,in figure 11 the error is lower, and the same happens in figure 12 for most of the patches, however in figure 17 (for patch 4) the MSE for $p = 4$ is slightly lower than for $p = 5$, and a simpler model should be selected. In spite of focusing our study on $p \leq 5$, we show this figure for higher polynomial orders and conclude that the optimal complexity for this specific data set is reached for $p = 10$. For higher polynomial degrees the variance and the error increases notably.

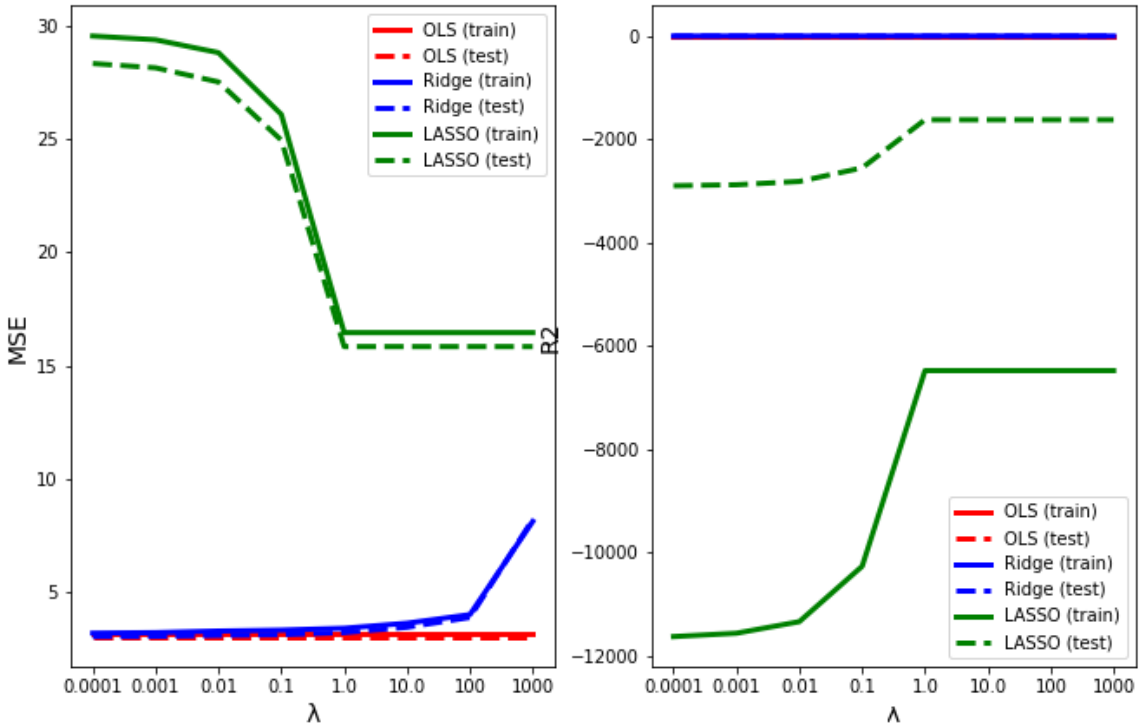

Figure 15: Performance of the OLS, Ridge and LASSO regressions on the SRTM image (patch 4) for different penalisation parameters (logarithmic axis) and $p = 5$. This figure was obtaining by resampling the data set 100 times. For more details, see notebook file.

Franke	**lower ci**	**upper ci**
β_0	0.2580	0.2885
β_1	8.1824	8.48720
β_2	4.5270	4.8312
β_3	-34.8050	-33.4281
β_4	-18.7201	-17.5029
β_5	-11.6934	-10.2170
β_6	42.2751	45.4831
β_7	48.5503	50.8878
β_8	27.0148	29.4662
β_9	-9.0479	-5.5154
β_{10}	-17.4951	-13.9934
β_{11}	-61.0595	-58.6679
β_{12}	-11.2307	-8.6275
β_{13}	-41.9344	-39.6184
β_{14}	30.6328	34.4655
β_{15}	-3.3440	-1.9155
β_{16}	21.3438	22.4165
β_{17}	10.9754	12.0818
β_{18}	-5.2443	-4.0581
β_{19}	21.7865	22.8209
β_{20}	-19.9274	-18.4005

Table 3: 95% confidence intervals for OLS with $p = 5$ based on self generated data.

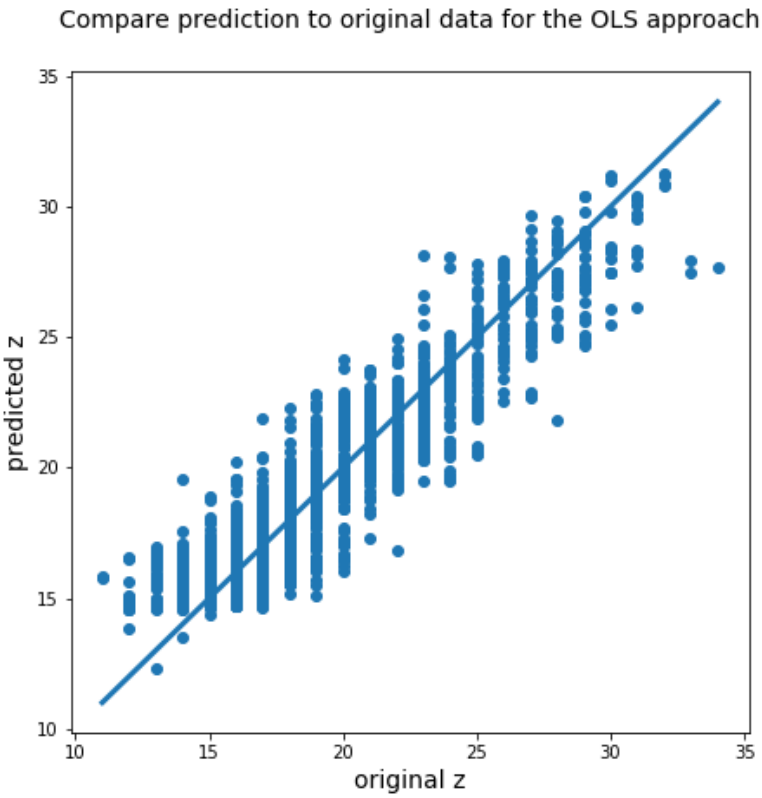

Figure 16: Scatter plot of the predicted test data and the original test points for the SRTM image and 100 resamplings. The model used is the OLS with $p = 5$.

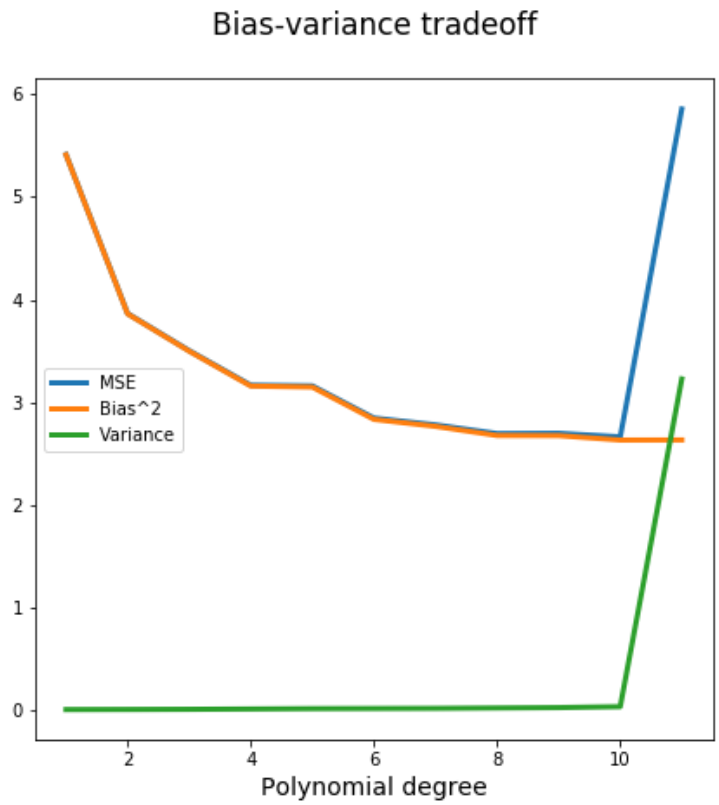

Figure 17: Bias-variance trade-off of the OLS applied to the SRTM image (patch 4). MSE, bias and variance are plotted as function of the polynomial degree. 100 resamplings were performed to generate this figure.

Conclusions

With a focus on the OLS, ridge and the LASSO regression analysis applied to terrain data, fitting polynomials of degree 5 and lower, we conclude that the model with the best performance on this data is the OLS with $p = 5$. This model does not reproduce all the characteristics of the terrain, meaning that it does not take advantage of the high resolution of the data, but proportionate a smoothed representation of the topography. The fact that the OLS performs better than the ridge and the LASSO means that a reduction of the variance by imposing a penalty on the regression coefficients does not reduce the error enough.

In a future work, we will apply LASSO to normalised data for higher polynomial orders, find the predictors which coefficient are not shrinked to zero, and then apply OLS with this new set of predictors. We will also explore a variety of methods for predicting surface elevation data, for instance, splines. Our aim is still to predict the surface that the SRTM do not cover. For this, we could also employ other observations as predictors, such as airborne imagery. Another possibility is to work with a larger set of images, grouping by similar terrains, in order to investigate whether the methods perform differently depending on the topography. We are aware that this is computationally expensive, so we might find some limitations.

In this study, we also compared our implementation of the methods at metrics used to the scikit-learn library, obtaining similar results, which demonstrates the simplicity of the methods used. The scikit-learn library was only used to test our code and for the LASSO regression.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical learning*. Springer-Verlag New York, New York 2009, pages 11-12, 44-45, 63-63, 68.

[2] B. Efron *Bootstrap Methods: Another Look at the Jackknife*. Ann. Statist., Volume 7, Number 1 (1979), 1-26.

[3] Franke, Richard. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Calhun. Monterey, California, 1979.

[4] Pankaj Mehta, Ching-Hao Wang, Alexandre G. R. Day, and Clint Richardson. *A high-bias, low-variance introduction to Machine Learning for physicists*[preprint].
`https://arxiv.org/pdf/1803.08823.pdf` [Accessed: 28.09.18].

[5] Raschka Sebastian, model evaluation and selection part 2
`https://sebastianraschka.com/blog/2016/model-evaluation-selection-part2.html`[Accessed: 28.09.18].

[6] Shuttle Radar Topography Mission
`https://www2.jpl.nasa.gov/srtm/` [Accessed: 28.09.18].

[7] USGS EROS Data Center
`https://earthexplorer.usgs.gov/` [Accessed: 28.09.18].

[8] Information about Montevideo, Wikipedia
`https://en.wikipedia.org/wiki/Montevideo/` [Accessed: 28.09.18].

[9] Machine Learning Crash Course: Part 4 - The Bias-Variance Dilemma
`https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/` [Accessed: 28.09.18].

[10] Hjort-Jensen, Morten. Linear regression and more advanced regression analysis
`https://compphysics.github.io/MachineLearning/doc/pub/Regression/pdf/Regression-minted.pdf`

[11] NASA. Shuttle radar topography mission - a mission to map the world.
`https://www2.jpl.nasa.gov/srtm/` [Assesed 08.10.18]

[12] Nancy Leiva Gutiérrez, Yolanda Rubiano Sanabria, Andrés Javier Peña Quiñones. (2015). Evaluation of soil moisture using topographic (DEM), climaten and soil parameters in piedmont area of villavicencio. *Revista EIA*. Vol 12. edition:Edición Especial N.2. Available at URL:
`http://www.scielo.org.co/pdf/eia/nspe2/nspe2a06.pdf`[Assesed 08.10.18]

[13] Tolentino PLM, Poortinga A, Kanamaru H, Keesstra S, Maroulis J, et al. (2016). Projected Impact of Climate Change on Hydrological Regimes in the Philippines. PLOS ONE 11(10): e0163941 Available at URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0163941 [Assesed 08.10.18]

[14] Arnold, D., Schicker, I. and Seibert, P.. High-Resolution Atmospheric Modelling in Complex Terrain for Future Climate Simulations(HiRmod). Report 2010. Available at URL: http://www.vsc.ac.at/fileadmin/user_upload/vsc/reports/BOKU-p70075-Schicker-Report.pdf [Assesed 08.10.18]

[15] Stuefer, M., Rott, H. and Skvarca, P. (2007) "Glaciar Perito Moreno, Patagonia: climate sensitivities and glacier characteristics preceding the 2003/04 and 2005/06 damming events," Journal of Glaciology. Cambridge University Press, 53(180), pp. 3–16. Available at URL: https://www.cambridge.org/core/services/aop-cambridge-core/content/view/286FCF679A466B2C482676A24627FFEC/S0022143000201743a.pdf/glaciar_perito_moreno_patagonia_climate_sensitivities_and_glacier_characteristics_preceding_the_200304_and_200506_damming_events.pdf

[16] McGranahan, Gordon, Balk Deborah and Anderson Bridget (2007). The rising tide: assessing the risks of climate change and human settlements in low elevation coastal zones. *Environment and Urbanization* [online]. Vol19(1) pp.17-37. Available at URL: http://journals.sagepub.com/doi/pdf/10.1177/0956247807076960 [Assesed 08.10.18]

Appendix

All the figures of the report are available in the folder results. Link to results folder.

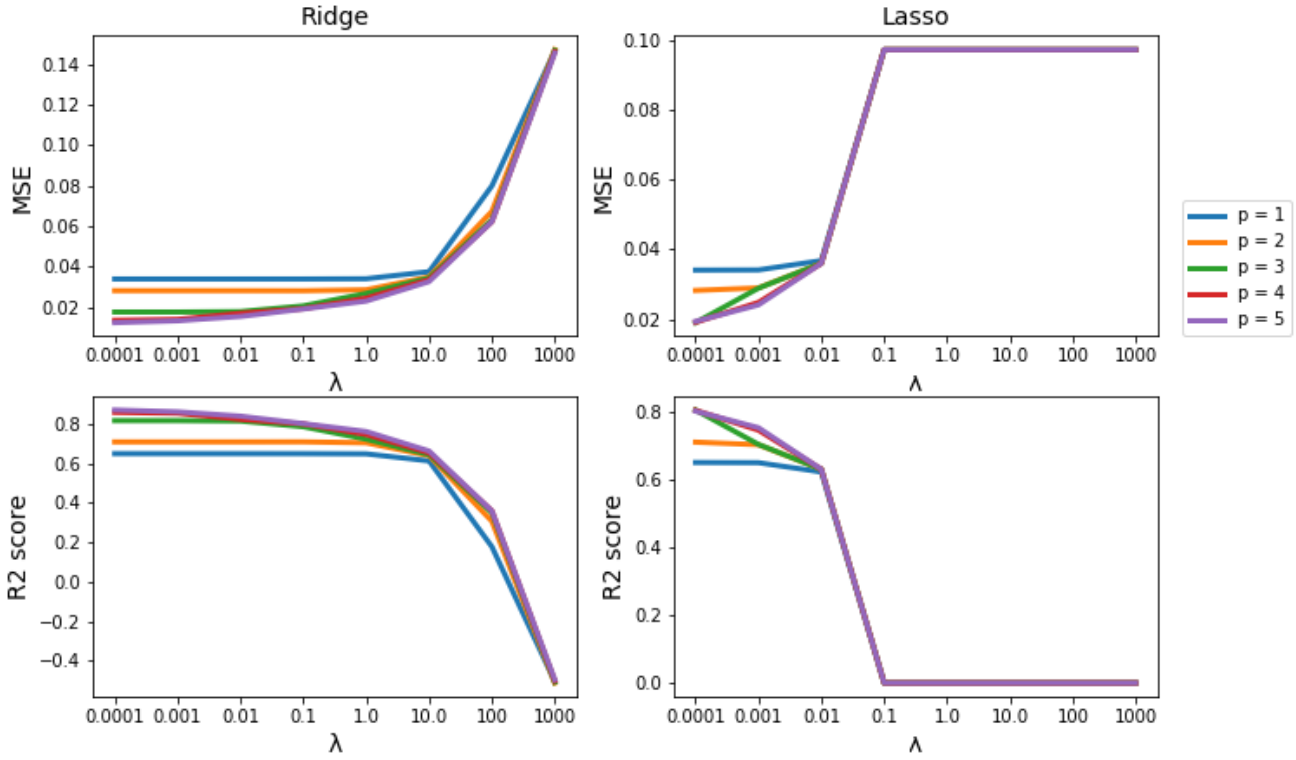

Figure 18: MSE score as a function of p for the OLS approach applied to the Franke's function. The different lines represent the random patches selected from the image. Patch number 4 is the selected for figure number 10. The same as figure 7, but for higher regularization parameters.

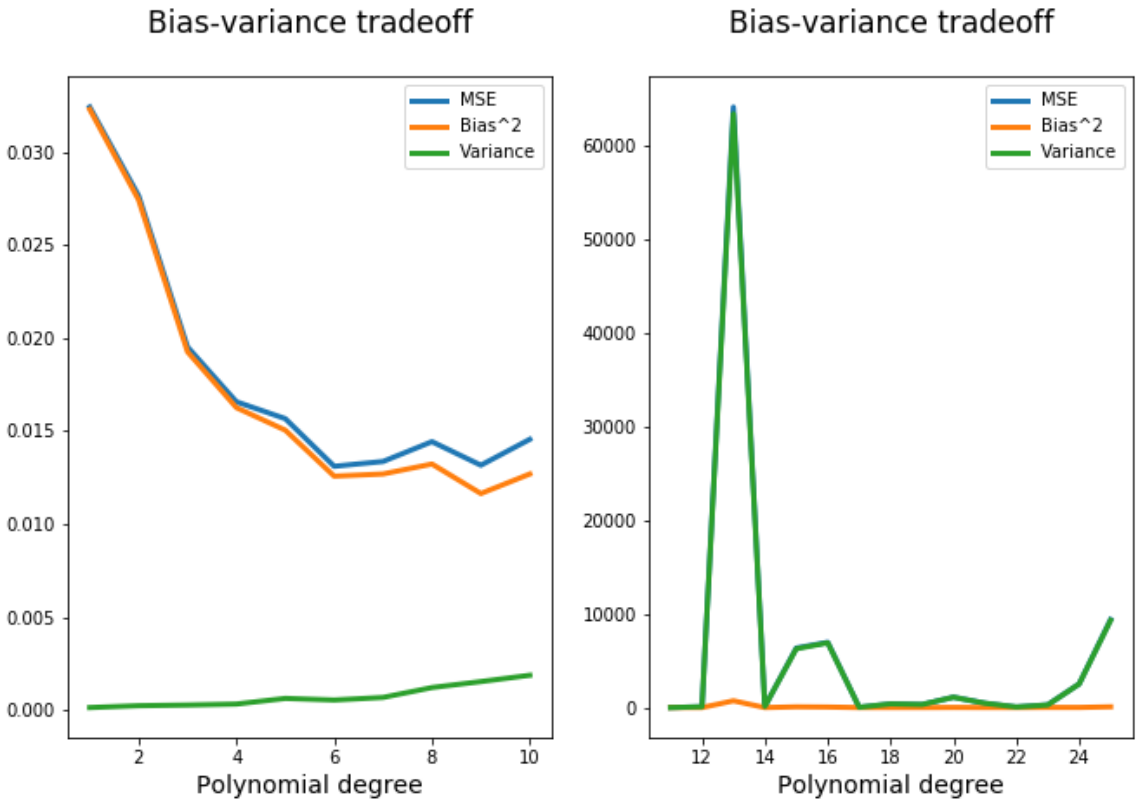

Figure 19: Bias-variance trade-off of the OLS applied to the Franke's function. MSE, bias and variance are plotted as function of the polynomial degree. 100 resamplings were performed to generate this figure. The same as figure 11 but for higher values of p.

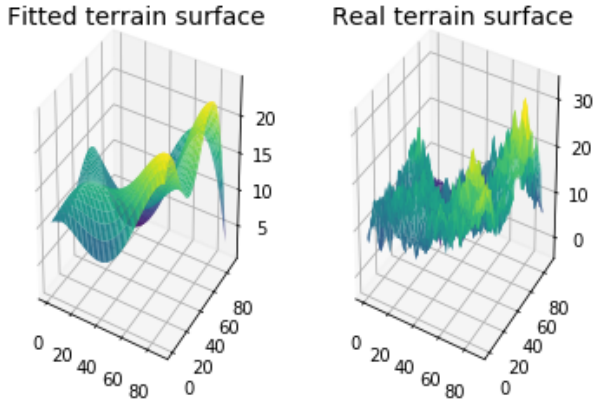

Figure 20: Fitted and real terrain surfaces of random patch of the image of Montevideo. The model used is the OLS for $p = 5$. This will be referred to as pathc number 0.

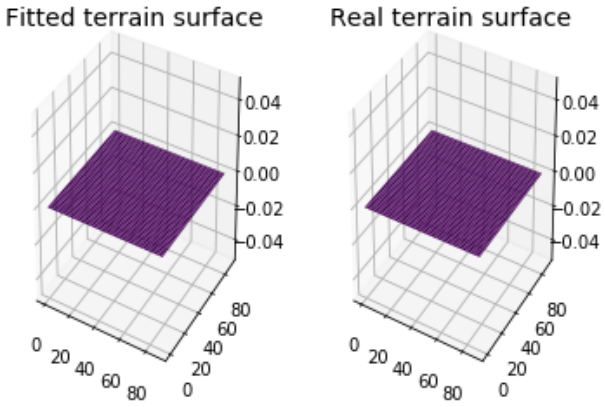

Figure 21: Fitted and real terrain surfaces of random patch of the image of Montevideo. The model used is the OLS for $p = 5$. This will be referred to as pathc number 1.

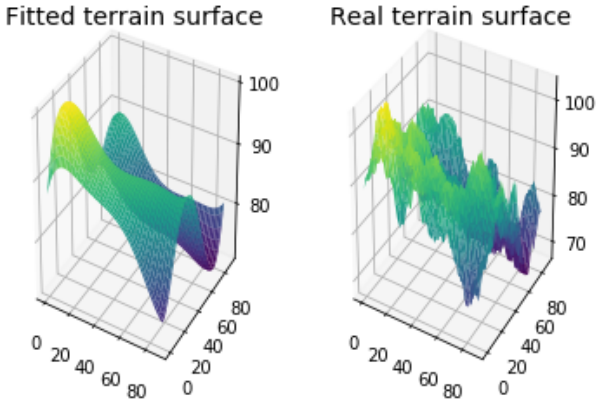

Figure 22: Fitted and real terrain surfaces of random patch of the image of Montevideo. The model used is the OLS for $p = 5$. This will be referred to as pathc number 2.

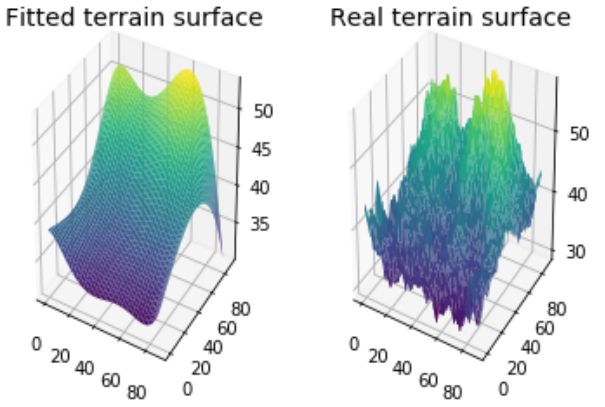

Figure 23: Fitted and real terrain surfaces of random patch of the image of Montevideo. The model used is the OLS for $p = 5$. This will be referred to as pathc number 3.

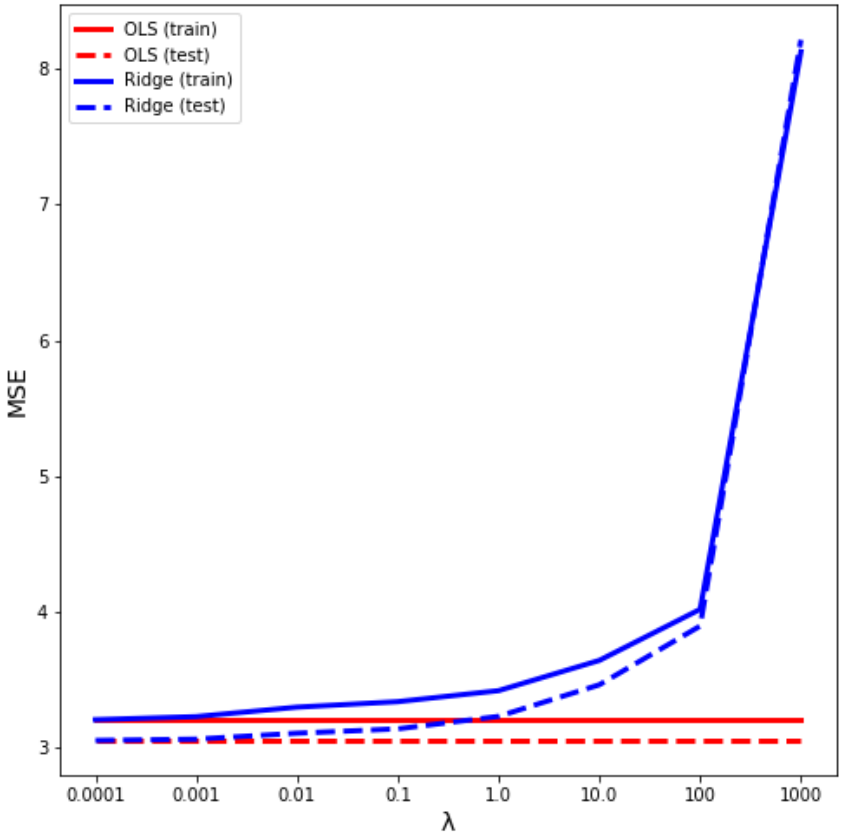

Figure 24: Performance of the OLS, Ridge and LASSO regressions on the SRTM image (patch 4) for different penalisation parameters (logarithmic axis) and $p = 5$. This figure was obtaining by resampling the data set 100 times. For more details, see notebook file. This is a zoom of figure 15 for OLS and ridge.

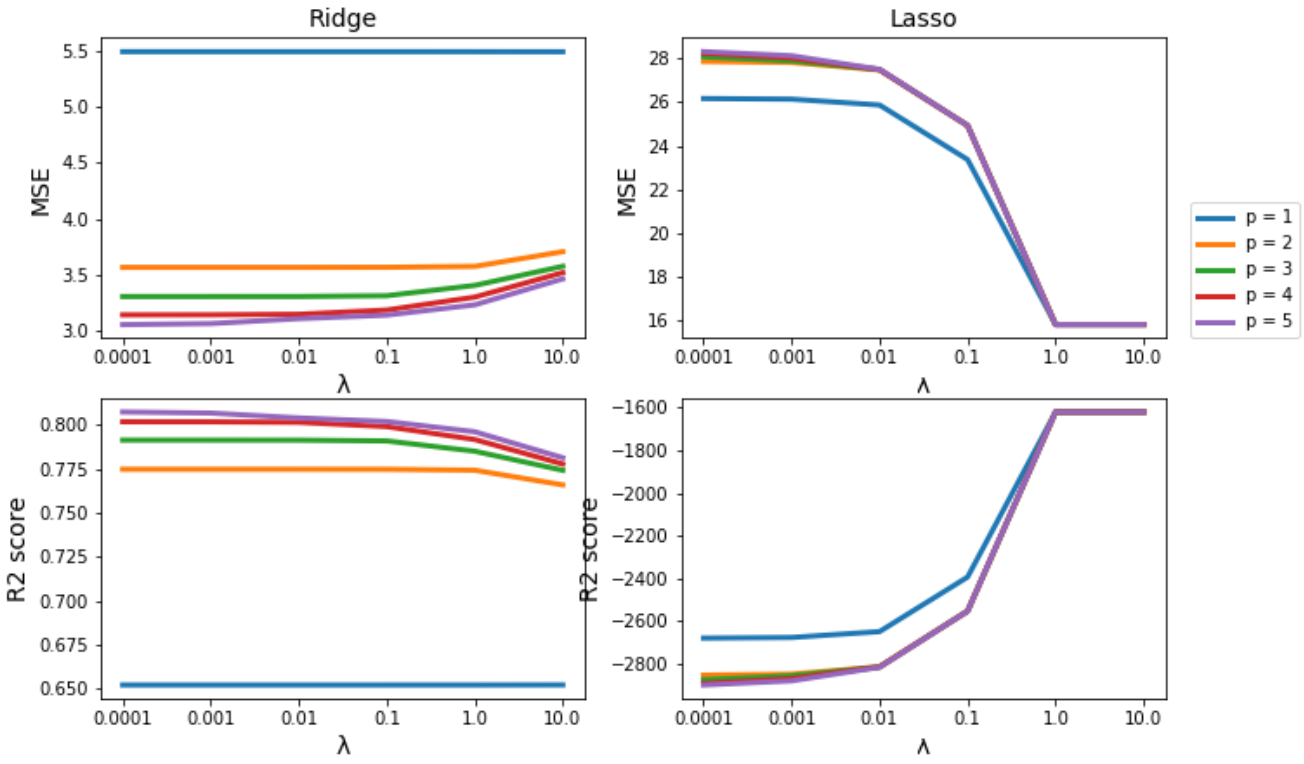

Figure 25: MSE score as a function of p for the OLS approach applied to the SRTM image, patch 4. This a zoom to figure 18.