Project 1

Data Analysis and Machine Learning FSY-STK4155

Maria Beatrice Cattini
Matteo D'Alessandro
Lisa Incollingo
Vanessa Ventura

October 7, 2022

Abstract

This report deals with the analysis of linear regression models, specifically Ordinary least squares, Ridge and Lasso. The analysis will be based on two types of data, one created as points on the Franke function's surface, and another by a GeoTIF picture containing the altitudes of a region near Stavanger in Norway. The models aim to learn from the training data and predict the output response: in order to do this, different model complexities have been explored, in the form of increasing polynomial degrees. The mean square error and the coefficient of determination (R^2-score) have been implemented, in order to assess the quality of these predictions and perform model selection. Furthermore, pre-processing methods, such as scaling, splitting, bootstrapping, and k-folds cross-validation, have been applied. We have performed bias-variance performance analysis on the models considered, in order to highlight possible regions of high bias or variance, and provide useful information for picking the most suitable model, which considering both vanilla and real data, turns out to be OLS one.

Contents

1 Introduction

Nowadays, huge amounts of data are continuously produced all over the world. This can be a great asset, but also poses a difficult challenge: how is it possible to transform data in real information? And how could we manage to find a way to describe and predict social, economic or natural phenomena behaviour from them?

We have to find a function, more precisely a model, that can do this flexibly but also accurately.

Hence, in this project, we are about to make our first steps through linear regression, resampling techniques and the analysis of bias-variance trade-off, comparing different models in order to find the best one.

Our work is structured in three principal parts: in section [2] we discuss about the methods we make use of in our analysis. Then, in section [3] we will apply what is treated before, and in the last section [4] we will consider the conclusion and possible improvements.

If we take a look in particular to the section [3], we will consider ordinary least squares regression, from now on referred to as OLS Regression [3.1], to construct a polynomial fit of the two-dimensional Franke function.

The Franke Function is defined as follows:

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

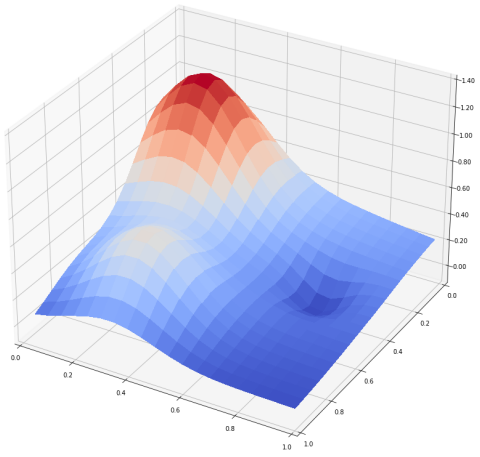

Figure 1: Franke Function Plot
We consider x and y values between 0 and 1

Then, we will implement two commonly used resampling techniques [3.3] the bootstrap method and the k-fold cross validation. Before applying them to our data we will always split our data into a training and a test set, training the linear models on the former and test their performance on the latter. We will also study the bias-variance trade-off [3.2] associated to each method, as function of the polynomial degree and the number of points in the dataset.

Then, we will do the same with the Ridge and the Lasso regression [3.4, 3.5, 3.4.2, 3.5.2] focusing on the performance as function of a penalty parameter λ, and make a comparison of the three regression methods in terms of MSE.

In the end we will analyse real 3.7, digital terrain data [1].

To obtain the results and plot the graphs included in this report, we made use of Python libraries such as **numpy**, **sklearn** and **matplotlib**.
The jupyter notebook used can be found on the GitHub [2].

[1] Dataset downloaded from: `https://earthexplorer.usgs.gov/`

[2] `https://github.com/matteodales/FYS-STK4155_Applied_Data_Analysis_and_Machine_Learning`

2 Method

2.1 Linear Regression

Regression analysis aims to explain the response or dependent variable y as function of a set of variables called predictors, explanatory variables or independent variables X. This can be done through a functional relationship: $y_i = f(X_{i*})$ but the form of this function is not always known, hence it is common to make the assumption of linearity between the two variables; in this case, the model is referred to as Linear Regression.
The response can be written in matrix form as:

$$y = X\beta + \varepsilon$$

where:
$\mathbf{y}$ is a vector with dimension $n \times 1$: $\mathbf{y} = [y_0, y_1, \ldots, y_{n-1}]^T$
β is a vector with dimension $(p+1) \times 1$: $\beta = [\beta_0, \beta_1, \ldots, \beta_{p-1}]^T$
ε is a vector with dimension $n \times 1$ of random error variables: $\varepsilon = [\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{n-1}]^T$

$\mathbf{X}$ is a design matrix with dimension $n \times (p+1)$: $\mathbf{X} = \begin{bmatrix} 1 & x_0^0 & x_0^1 & \cdots & x_0^{p-1} \\ 1 & x_1^0 & x_1^1 & \cdots & x_1^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1}^0 & x_{n-1}^1 & \cdots & x_{n-1}^{p-1} \end{bmatrix}$

We introduce the following hypothesis for model:

a) $\mathbf{X}$ deterministic matrix with full-rank: $rank(\mathbf{X}) = p + 1 < n$
b) ε_i are independent random variables with:
   - $\mathbb{E}[\varepsilon_i] = 0$
   - $Var[\varepsilon_i] = \sigma^2$
   - $\varepsilon_i \sim N(0, \sigma^2)$

Starting for these hypothesis, we can prove the following statements:

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

$$\begin{aligned} \mathbb{E}[y_i] &= \mathbb{E}\left[\sum_{j=0}^{p-1} x_{ij}\beta_j + \varepsilon_i\right] \\ &= \mathbb{E}\left[\sum_{j=0}^{p-1} x_{ij}\beta_j\right] + \underbrace{\mathbb{E}\left[\varepsilon_i\right]}_{\varepsilon \sim N(0,\sigma_\varepsilon^2)} \\ &= \mathbb{E}\left[\sum_{j=0}^{p-1} x_{ij}\beta_j\right] + 0 \\ &= \mathbb{E}\left[\mathbf{X}_{i*}\boldsymbol{\beta}\right] + 0 \\ &= \mathbf{X}_{i*}\boldsymbol{\beta} \end{aligned}$$

b) $\mathrm{Var}[y_i] = \sigma^2$
$\mathrm{Var}[y_i] = \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2$, where $y_i = \mathbf{X}_{i*}\boldsymbol{\beta} + \varepsilon_i$

$$\begin{aligned}
\mathbb{E}[y_i^2] &= \mathbb{E}[(\mathbf{X}_{i*}\boldsymbol{\beta} + \varepsilon_i)^2] \\
&= \mathbb{E}[\mathbf{X}_{i*}\boldsymbol{\beta}^2 + \varepsilon_i^2 + 2\mathbf{X}_{i*}\boldsymbol{\beta}\varepsilon_i] \\
&= \mathbf{X}_{i*}\boldsymbol{\beta}^2 + \underbrace{\mathbb{E}[\varepsilon_i^2]}_{\sigma_\varepsilon^2} + 2\mathbf{X}_{i*}\underbrace{\mathbb{E}[\varepsilon_i]}_{0} \\
&= \mathbf{X}_{i*}\boldsymbol{\beta}^2 + \sigma_\varepsilon^2 + 0
\end{aligned}$$

$$\mathrm{Var}[y_i] = \mathbf{X}_{i_*}\boldsymbol{\beta}^2 + \sigma_\varepsilon^2 - \mathbf{X}_{i_*}\boldsymbol{\beta}^2 = \sigma_\varepsilon^2$$

The linear regression coefficients β_j contained in the vector β have to be estimated. In order to find the optimal parameters, the ordinary least square techniques is usually used.

2.1.1 Ordinary least square (OLS)

OLS is a method of determining the regression coefficients by minimizing the cost function:

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

In a more compact way we can use the design matrix $\mathbf{X}$ and introducing the definition $\tilde{y}$ as $\tilde{y} = \mathbf{X}\beta$, hence we can write

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\big\{(y - \mathbf{X}\beta)^T(y - \mathbf{X}\beta)\big\}$$

The OLS estimates $\hat{\beta}$ for β are the coefficients which make the sum of squares of the residuals as low as possible. Hence, in order to find the parameters β_j we have to minimize the cost function, by putting its derivative equal to 0.

$$\frac{\partial C(\beta)}{\partial \beta} = 0$$

By solving the equation we find: $\mathbf{X^T}y = \mathbf{X^T}\mathbf{X}\beta$ and given the hypothesis a), the matrix $\mathbf{X^T}\mathbf{X}$ is invertible. Thus we have the estimation of β:

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

We can prove that the $\hat{\beta}^{OLS}$ estimator is *unbiased*, meaning that its expectation is equal to the real parameter:

$$\begin{aligned}\mathbb{E}[\hat{\beta}] &= \mathbb{E}[(\mathbf{X^T X})^{-1}\mathbf{X^T y}] \\ &= (\mathbf{X^T X})^{-1}\mathbf{X^T}\mathbb{E}[y] \\ &= \underbrace{(\mathbf{X^T X})^{-1}\mathbf{X^T X}}_{I}\beta \\ &= \beta\end{aligned}$$

2.1.2 Variance of the parameters β

We have seen that the observations y_i are normally distributed and uncorrelated, have variance σ^2 and that X_i are deterministic. Now, we can show that: $\mathrm{Var}[\hat{\beta}] = \sigma^2(\mathbf{X^T X})^{-1}$

$$\begin{aligned}\mathrm{Var}[\hat{\beta}] = \mathrm{Var}[(X^T X)^{-1}X^T y] &= (\mathbf{X^T X})^{-1}\mathbf{X^T}\sigma_\varepsilon^2\mathbf{X}(\mathbf{X^T X})^{-1} \\ &= \sigma_\varepsilon^2\underbrace{(\mathbf{X^T X})^{-1}\mathbf{X^T X}}_{I}(\mathbf{X^T X})^{-1} \\ &= \sigma_\varepsilon^2(\mathbf{X^T X})^{-1}\end{aligned}$$

We can use this result for define the confidence interval for the parameters β:

$$C.I. : [\hat{\beta} - z_{1-\alpha}(\mathbf{X^T X})^{-1}\sigma^2,\ \hat{\beta} + z_{1-\alpha}(\mathbf{X^T X})^{-1}\sigma^2]$$

2.1.3 Accuracy measurement

Evaluating the model accuracy is an essential part of the process of creating machine learning models, since it allows us to understand how well the model is performing in its predictions. The basic concept of accuracy evaluation is to compare the original target with the predicted one according to a certain measurement. In the following analysis, two of these accuracy measurements will be used: MSE and R^2.

Mean Square Error

The MSE measures the square difference between the predicted values and the real ones.
We can calculate the value by the formula:

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

The measure is always non negative. Smaller values of error are preferable, as a smaller difference generally implies a better fit.

One of the disadvantages of using the MSE is that its value is not scale-invariant: it varies based on whether the values of response variable have been scaled or not. More details in scaling data are given in section [2.5]

Coefficient of determination: R^2

R^2 represents the proportion of the variance in the dependent variable which is explained by the linear regression model. We can write R^2 as:

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

Considering that correlation measures the strength of the relationship between the dependent variable and the independent, the R^2 explains to what extent the variance of one variable explains the variance of the second variable.
R^2 is always between 0 and 1:

- 0 represents a pointless model that does not explain any of the variation of the response variable. The mean of the dependent variable predicts the dependent variable as well as the regression model.

- 1 represents a model that explains all the variation of the response variable.

2.2 Shrinkage methods

We have just seen the OLS regression, which allows us to find optimal parameter $\hat{\beta}$ which, as we have seen, is *unbiased*.
This is not the only way to set up a regression: the other methods we are going to consider are Ridge and Lasso regression, so called shrinkage methods. These methods are not unbiased, but they have important consequences on the flexibility prediction ability and bias-variance trade-off.

2.2.1 Ridge Regression

If we add a regularization parameter λ to the design matrix X we obtain the follow cost function:

$$C(X, \beta) = \frac{1}{n}||y - \mathbf{X}\beta||_2^2 + \lambda||\beta||_2^2$$

in which $||\beta||_2$ is a norm vector $||\beta||_2 = \sqrt{\sum_{j=0}^{p-1}\beta_j^2}$.
It can be shown that the introduction of this penalty term is equivalent to requiring that $||\beta||_2^2 < t$, where t is a finite number larger than 0.
Solving the Ridge optimization equation, we find that:

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

which corresponds to $\hat{\beta}^{OLS}$ modified by a diagonal term added to $\mathbf{X^T X}$.
By increasing the value of λ, the penalty term becomes more and more prevalent

in the cost function and the values for β tend to be shrunk towards zero.
The penalty term introduces a dependency of the method on the scale of the data, since the squared norms of the coefficients are used to compute the cost function: for this reason, when implementing Ridge regression, considerations about scaling are crucial.

Ridge Regression linked to OLS Regression

Assuming true the hypothesis a) and b) [3] we can prove that:

$$\mathbb{E}\big[\hat{\beta}^{Ridge}\big] = (\mathbf{X^T X} + \lambda \mathbf{I})^{-1}(\mathbf{X^T X})\hat{\beta}^{OLS}$$

From that we obtain that $\mathbb{E}\big[\hat{\beta}^{Ridge}\big] \neq \hat{\beta}^{OLS}$ for any $\lambda > 0$. So $\hat{\beta}^{Ridge}$ is a biased estimator of β.
We could also prove that:

$$\mathrm{Var}\big[\hat{\beta}^{Ridge}\big] = \sigma^2\big[\mathbf{X^T X} + \lambda \mathbf{I}\big]^{-1}\mathbf{X^T X X^T X} + \lambda \mathbf{I}]^{-1^T}$$

and that $\mathrm{Var}\big[\hat{\beta}^{OLS}\big] > \mathrm{Var}\big[\hat{\beta}^{Ridge}\big]$ for $\lambda > 0$.
So $\hat{\beta}^{Ridge}$ gives a biased estimator of β but with a lower variance than $\hat{\beta}^{OLS}$, hence gives a more restrictive range in which the estimated parameters vary.

2.2.2 Lasso Regression

Lasso stands for "least absolute shrinkage and selection operator". We define now the Lasso cost function:

$$C(\mathbf{X}\beta) = \frac{1}{n}||y - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1$$

in which we define norm-1 as:

$$||\mathbf{X}||_2 = \sum_i |\mathbf{X}_i|$$

The analytical equation for Lasso Regression coefficients is not as simple as others to represent, since the penalty term, being a 1-norm, is not differentiable. The Lasso regression function doesn't shrink β parameters to 0 continuously, but rather forces certain coefficients to zero as the λ value increases. This means that the less significant explanatory variables are cut off from the model, hence the Lasso is a way to perform variable selection, as its name states.

[3] See point 2.1 above

2.3 Bias-variance trade-off

Our main goal is to estimate regression coefficients in such a way as to minimize errors. The bias-variance trade-off plays an important role in this optimization. In order to find the parameters β we optimize the mean squared error with the the cost function:

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

If we focus on this equation we can then note that it is possible to rewrite it in terms of variance of the model,bias and irreducible error, as follows below.

$$\mathbb{E}[(y - \tilde{y})^2] \text{ where } y = f(x) + \varepsilon$$

In order to make it simpler, we write $f(x)$ as f. Hence, $\mathbb{E}[(f + \varepsilon - \tilde{y})^2]$

We can now add and subtract $\mathbb{E}[\tilde{y}]$:

$$\mathbb{E}[(f - \varepsilon - \tilde{y} + \mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}])^2] =$$

By solving the square:

$$= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2 + (\tilde{y} - \mathbb{E}[\tilde{y}])^2 + \varepsilon^2] =$$
$$= \tfrac{1}{n}\textstyle\sum(f_i - \mathbb{E}[\tilde{y}])^2 + \underbrace{\frac{1}{n}\sum(\tilde{y}_i - \mathbb{E}[\tilde{y}])^2}_{\text{var}(\tilde{f})} + \underbrace{\mathbb{E}[\varepsilon^2]}_{\sigma^2} =$$
$$= \underbrace{\frac{1}{n}\sum(y_i - \mathbb{E}[\tilde{y}])^2}_{(\text{Bias}[\tilde{y}])^2} + \text{var}(\tilde{f}) + \sigma^2$$

The first term $\frac{1}{n}\sum(y_i - \mathbb{E}[\tilde{y}])^2$ is the bias: the difference between the average prediction of our model and the correct value which we are trying to predict. The second term $\text{var}(\tilde{f})$ is the variance of the model that we have chosen, which represents the variability of the model prediction for a given datapoint: models

with high variance do not generalize well on previously unseen data. The last term σ^2 of this equation is the irreducible error, which is linked to the nature of the data and is not influenced by the chosen model. No matter how good our model is, the data will always have a certain quantity of "noise" that can't be reduced.

The ideal situation would be, of course, the one with a low value of both the bias and the variance of the model. But we can run into several cases as shown in the picture below:

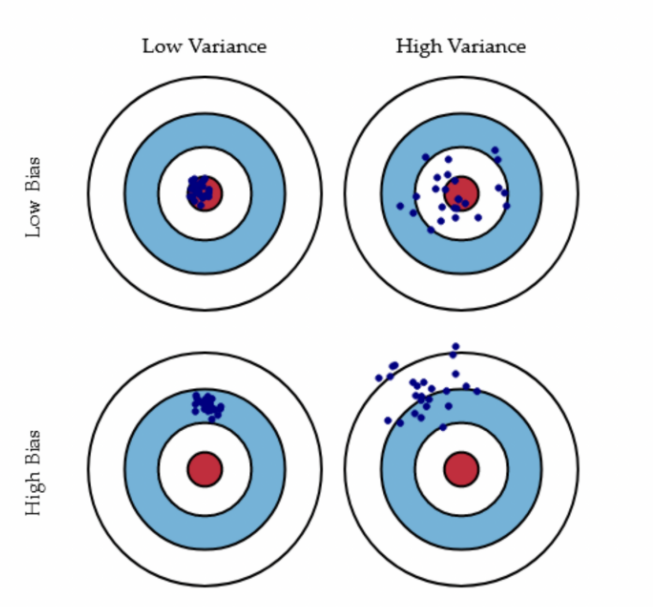

Figure 2: Variance trade off

In order to have good prediction we have to find a balance between the two quantities that we can control. A more complex model will usually present lower bias, since the model can better adapt to the data. On the other hand, with increasing complexity, the variance tends to increase. With less complexity we would expect lower variance, since the the model wouldn't change that much by changing the data, but the bias would be high.

2.4 Resampling techniques

Our dataset is only a sample belonging to a bigger population, so the values we calculate are only sample mean, variance and others. If we could ideally have as much data as possible, we will obtain an estimate that is near to the true value of the distribution, but it's not possible in most of the cases. One of the ways to solve the problem is the use of resampling techniques, whose specific aim is to

improve the accuracy of our estimated values working only on our dataset. We now introduce the k-fold cross validation and bootstrap method.

2.4.1 K-fold cross validation

The k-fold cross validation algorithm is implemented as follows.
For the various values of the data set:

1. Shuffle the dataset randomly
2. Split the dataset into k groups, generally 5-10 groups
3. For each unique group:
   a) Take the group as a hold out or test data set
   b) Take remaining groups as a training data set
   c) Fit a model on the training set and evaluate it on the test set
   d) Retain the evaluation score and discard the model. We focus on MSE as the evaluation score.
4. Summarize the skill of the model using the sample of model evaluation scores. We calculated the average of every MSE as a score of the model prediction performance.

The parameter k refers to the number of groups that a given data sample is to be split into. There is not a given rule in what should be the value of k, but usually it is 5 or 10. Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

Figure 3: The figure shows the 5-fold cross validation algorithm.

2.4.2 Bootstrap

The Bootstrap method consists of resampling the datapoints in the dataset with repetition to create a certain number of bootstrap samples as training sets.

The algorithm follows these steps:

1. Split the dataset into two sets: training set and test set.
2. Draw a bootstrap sample from the training set with reimmission, whose size is the same as the original training set.
3. Fix a number of bootstrap samples. For each bootstrap sample:
   a) Retain the bootstrap sample as the training set
   b) Fit the model on the training set and evaluate it on the test set.
   c) Retain the evaluation score and discard the model. We keep MSE as the evaluation score.
4. Summarize the skill of the model using the sample of model evaluation scores. We calculated the average of every MSE as a score of the model prediction performance.

2.5 Scaling data

Before starting to work on our data it's important to have a look at their magnitude and scale. Differences in the scales across input variables may increase the difficulty of the problem being modelled.
An example of this is that large input values (e.g. a spread of hundreds or thousands of units) can result in a model that learns large weight values. A model with large weight values is often unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalisation error. In particular, the variables are required to be a common scale in order to apply a penalty, as in the case of Lasso and Ridge regression: since the applied penalties are not scale invariant, it's important that the variables involved have a common scale.
Some of the most common scaling methods are:

- Standardization
$$x_{scaled} = \frac{x - \bar{x}}{\sigma_x}$$
- Min-Max scaling
$$x_{scaled} = \frac{x - min_x}{max_x - min_x}$$

3 Discussion and Results

3.1 OLS Regression analysis

In this chapter we focus on studying how to fit polynomials to a specific two-dimensional function called Franke function. After having implemented our methods, we will employ resampling techniques such as cross-validation and bootstrapping to accurately test the results.

3.1.1 Generating the dataset

The data studied in this section is obtained randomly. After deciding on a number of points *npoints*, two vectors of length *npoints* containing values in the $[0, 1]$ interval corresponding to the x and y coordinates of the points are generated. The Franke's function is then computed to obtain the response variable we want to predict. In this section we chose a value of $npoints = 100$.
We also considered the addition of a noise term that is normally distributed with mean 0.

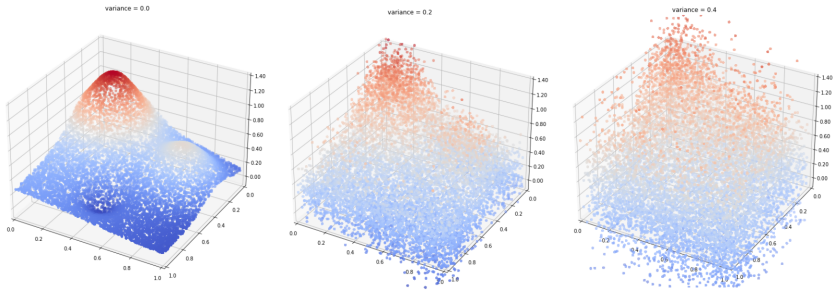

Figure 4: Franke function plot with different noise variances

Here, the three graphs show how the resulting function changes by changing the variance of the noise.

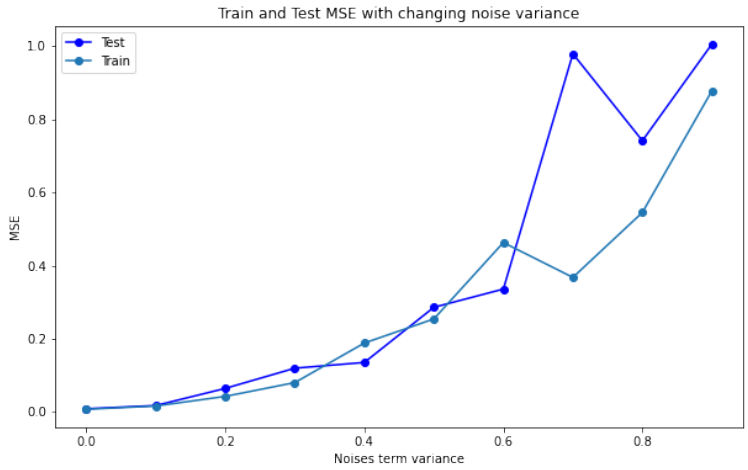

Figure 5: Train and Test MSE with changing noise variance

In Figure 5 we show the values of train and test MSE by changing the variance of the noise, that we will discuss more properly in section [3.1.3]. Increasing the noise of the data, the variance, and consequently the average error we obtain, increases.

Unless otherwise specified, the following results are obtained by introducing a noise of variance 0.1: we chose this value to obtain results that are not excessively influenced by the noise term, but still working with a challenging regression problem.

The generated data will not be scaled before the analysis: in fact, considering the $[0, 1]$ interval for both coordinates, and observing that the Franke function values are contained in the $[0, 1.3]$ interval for this area, the variables considered are distributed on very similar scales and the standardization is not necessary.

3.1.2 Fitting the model and prediction

As explained, the aim of this first part is to model the data that we have through a polynomial in x and y up to the fifth order.
In order to do this, we utilized the **PolynomialFeatures** function from the **sklearn.preprocessing** package, which allows us to transform the generated coordinates of the points in order to consider all polynomial terms related to a certain degree.
Finally, before fitting the model, it's necessary to divide our data into a test and training set: for this task, the **train_test_split** function from **sklearn.model_selection** package has been implemented.
Unless otherwise specified, the size of the test set in this and the following sections is 20% of the whole dataset.

3.1.3 Outputs and result

The following figures show how different polynomial surfaces can fit our dataset and the relative $\hat{\beta}^{OLS}$ coefficients.
How can we evaluate what the best option is?

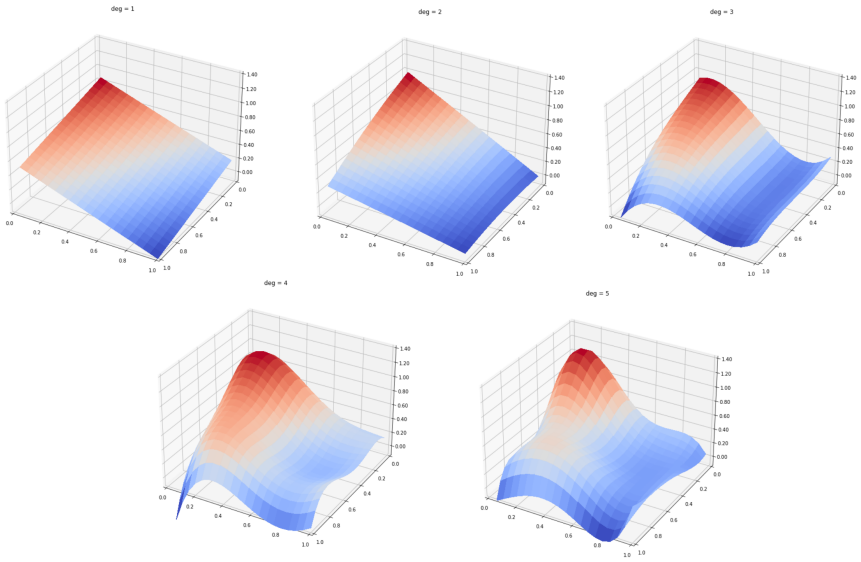

Figure 6: OLS fits of data took by Franke Function with polynomial of degrees 1, 2, 3, 4 and 5

	1	*2*	*3*	*4*	*5*
	β	β	β	β	β
1	0.925	1.049	0.843	0.459	-0.162
x	-0.329	-0.662	0.053	4.751	9.000
y	-0.725	-0.748	1.174	3.151	9.379
x^2		-0.126	-1.915	-20.136	-24.535
xy		0.808	1.542	-1.917	-38.077
y^2		-0.363	-5.843	-12.145	-24.623
x^3			0.694	25.870	14.464
x^2y			1.483	10.317	60.919
xy^2			-2.168	-3.805	79.310
y^3			4.299	13.729	7.075
x^4				-11.583	12.649
x^3y				-4.040	-41.266
x^2y^2				-3.007	-49.901
xy^3				3.246	-86.108
y^4				-5.041	29.746
x^5					-10.703
x^4y					2.867
x^3y^2					28.434
x^2y^3					3.225
xy^4					40.312
y^5					-21.931

Figure 7

In order to analyse the performance with different polynomial degrees we can use the accuracy measurements [2.1.3]: MSE and R^2.

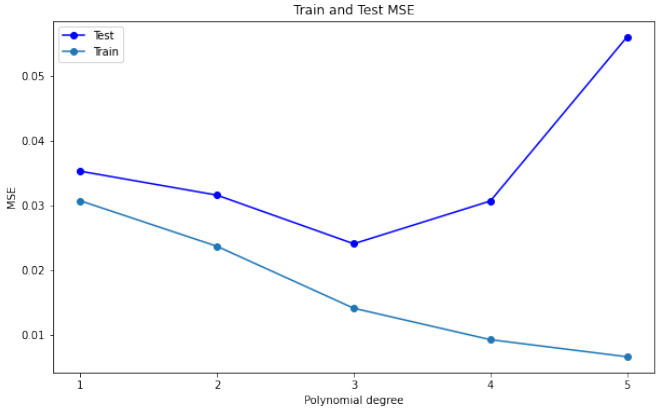

Figure 8: Train and Test MSE

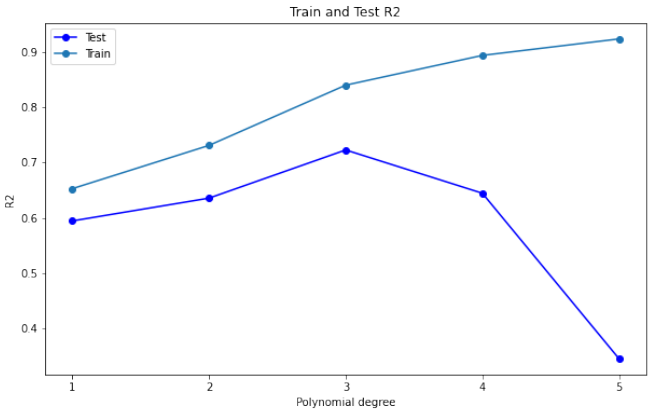

Figure 9: Train and Test R^2

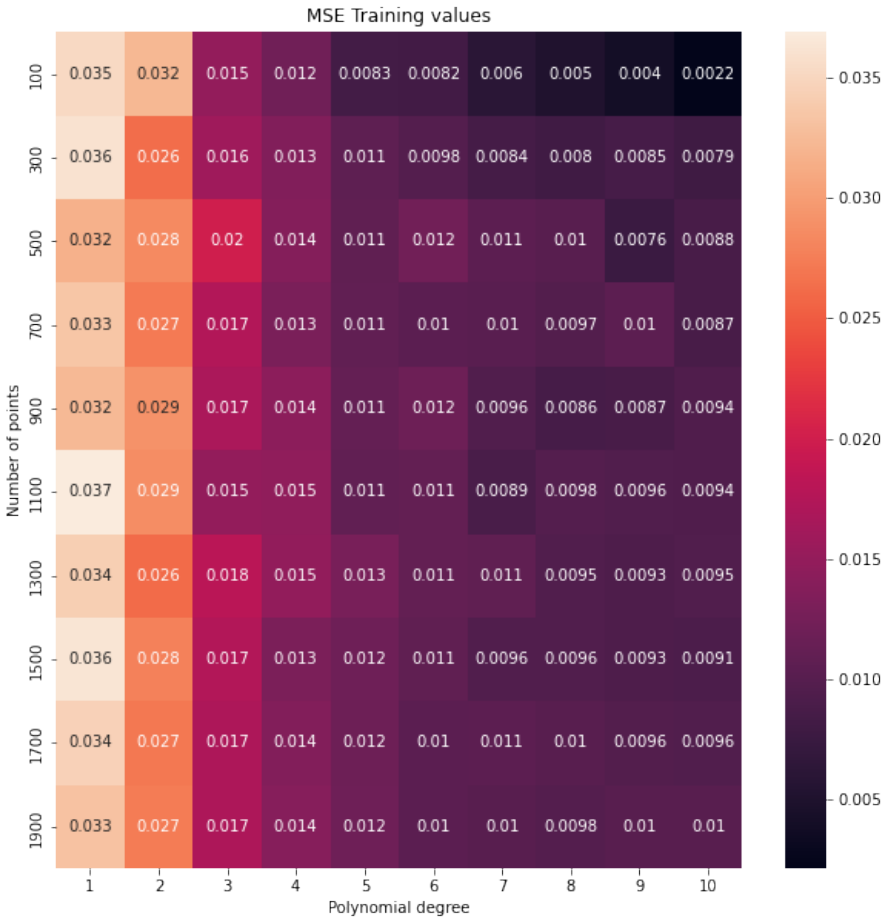

Figure 10: MSE training heatmap

The graphs above show the dependence of MSE and R^2 on the degree of the polynomial, divided in test and training.
Considering the training line, on Figure 5 and Figure 6, we can easily see that the MSE decreases as the polynomial degree increases, while the R^2 increases.
Differently, if we now focus on the testing line, the MSE after the third degree polynomial starts to grow rapidly, while the R^2 at the same degree, once it has reached a 0.7 value, starts to decrease. We can assume that the difference in the patterns between train and test sets is the result of overfitting.

The need for a distinction between test and training points is further analysed in Figure 7. Indeed, we can see that the best value of training MSE corresponds to the highest degree of the polynomial and the lowest number of points that we consider. The fewer data we have and the more the polynomial will be able to

accurately adapt to it, the more the prediction error will be contained and close to zero.
Of course, this result does not reflect in the quality of our model, and is once again the result of overfitting. In practice the MSE training is of little interest to us. What we are really concerned about is how well the model can predict values on new unseen data.

According on our results, the degree that best fits the data in our benchmark is the 3rd.

3.2 Focus on MSE and bias-variance trade-off

In this part of the study we want to explore in detail the relation between the test MSE and the bias-variance trade-off, through our results.
As previously observed, the test MSE [4] initially decreases as we increase the flexibility of the model but eventually starts to increase again. How can we explain this pattern?
By allowing the model to be extremely flexible we are letting it fit to patterns in the training data, but as soon as we introduce new unseen data points in the test, the model isn't able to generalise well because these patterns are only a property of the train data. We can run into a situation of overfitting.
This "u-shape" of the test MSE curve is related to the bias-variance trade off [2.3].

[4] Figure 5

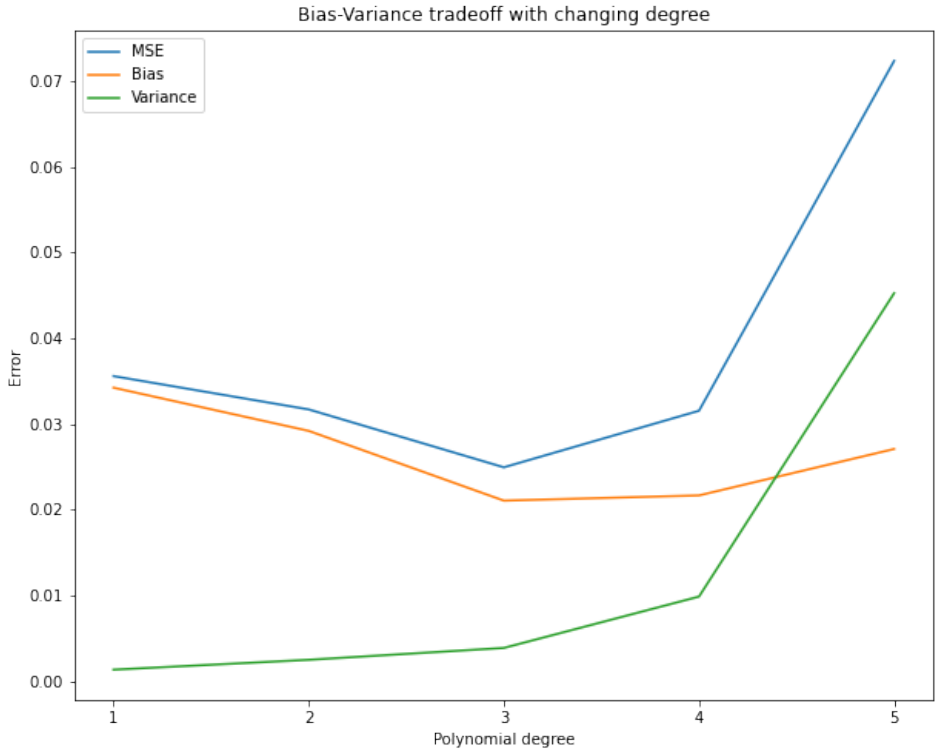

Figure 11: Bias-variance trade-off with changing degree.

In this graph we can see the bias-variance decomposition of the test MSE with changing degree of the polynomials, obtained using the bootstrap method.
In the left side of the graph we can see zone of high bias and low variance. The model is not able to capture the true underlying function behaviour.
For higher polynomial degrees, we can see that the bias decreases and the variance increases. A model with high variance suggests overfit to the training data, and results in high MSE test.
Finally, the graph highlights how the third degree presents the best balance between variance and bias.

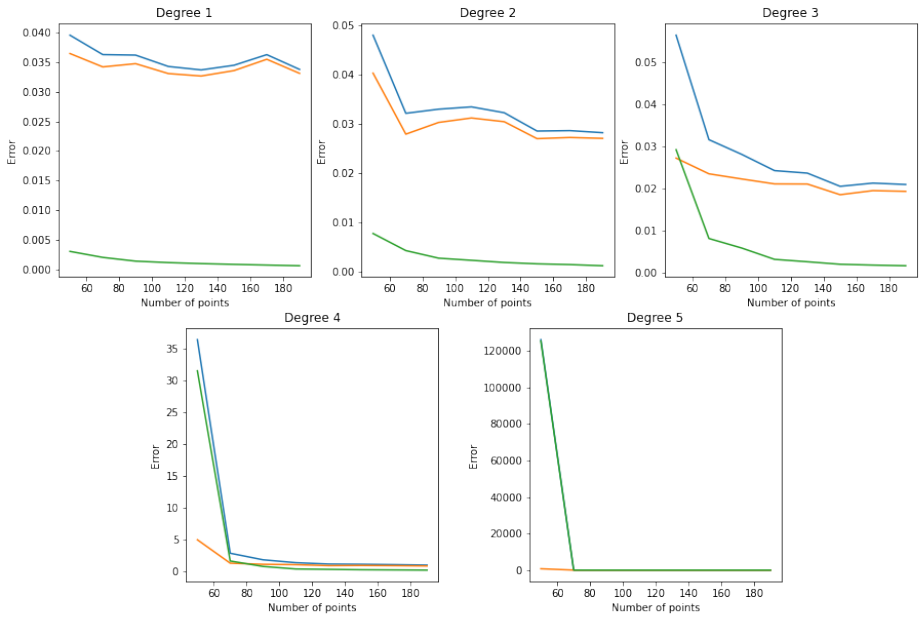

Figure 12: Bias-Variance trade off with changing degree and number of points

In Figure 9 we can see how the decomposition of the MSE test (in blue) changes by considering different polynomial degrees and different number of points. For lower degrees, the bias (represented by the orange line) is the main component of the MSE: the model has generally less variance but the underlying Franke function is not well captured. For increasing degrees, the variance (in green) becomes more and more prevalent, especially when paired with a low number of points.

3.3 OLS regression and resampling methods

In this section, we have compared the test error obtained by bootstrapping and cross-validation as a function of the polynomial degree.

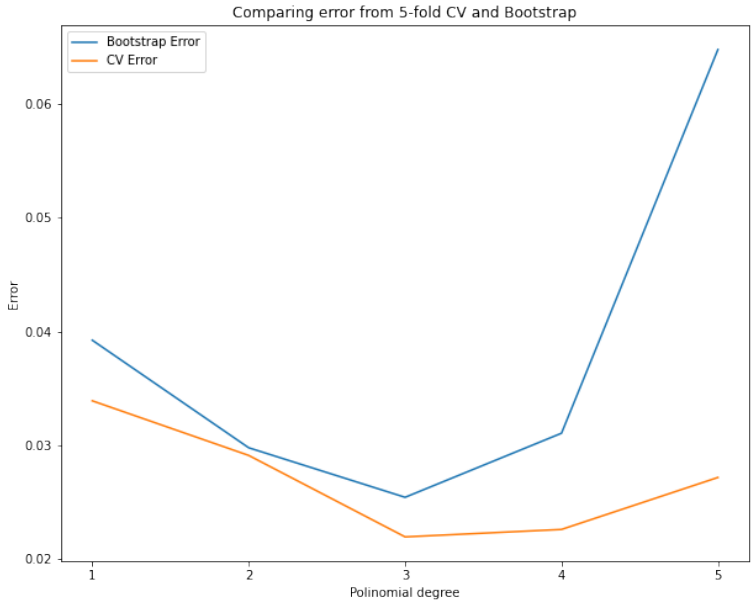

Figure 13: Comparing error from 5-fold CV and Bootstrap

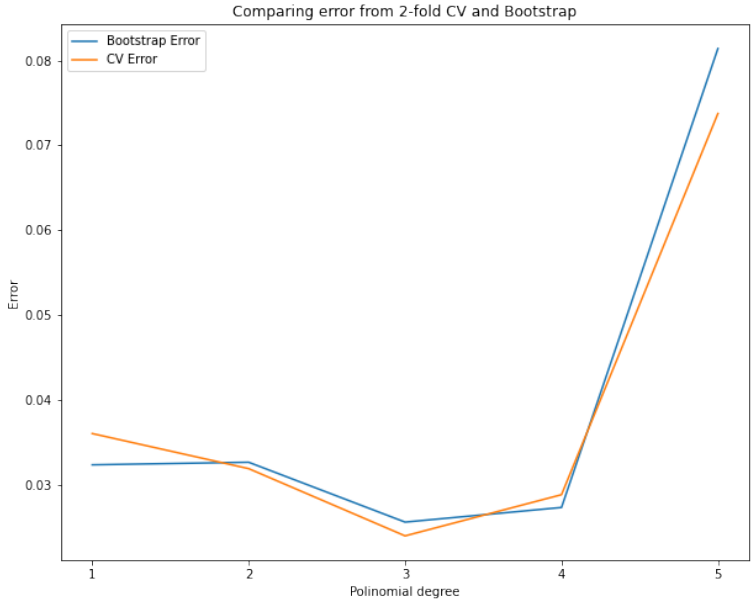

Figure 14: Comparing error from 2-fold CV and Bootstrap

The polynomial degree that corresponds to the smallest error is 3 in the 5-fold cross validation, as well as in the bootstrap.

Figure 12 compares the MSE values for bootstrapping with 100 total points, 20% of points used as a test set and 80 bootstrap iterations, and 5-fold cross validation. It's clearly visible that the CV error is generally lower, especially for higher degrees. This result can be explained by looking at the amount of data that each method actually uses for training: while 5-fold cross validation divides the total set of points in 5, effectively using 80% of data for training, the bootstrap method performs a resampling with repetition of the training set, causing the repetition of some data points and the exclusion of about 36% of points in the original set. This difference in the size of the training data for the two methods causes the bootstrapping model to fall into overfitting for lower polynomial degrees, leading to a higher error.

This explanation is supported by the Figure 13 : when we consider 2-fold CV, which consists of dividing the dataset in two, using 50% of data for training, the results on the MSE are much more similar.

The last graph shows the changing MSE curves for different numbers of folds in the cross-validation method. From 5 to 10 folds, the difference in the curves is minimal, and degrees 3 and 4 are clearly the best options for the OLS regression within our framework.

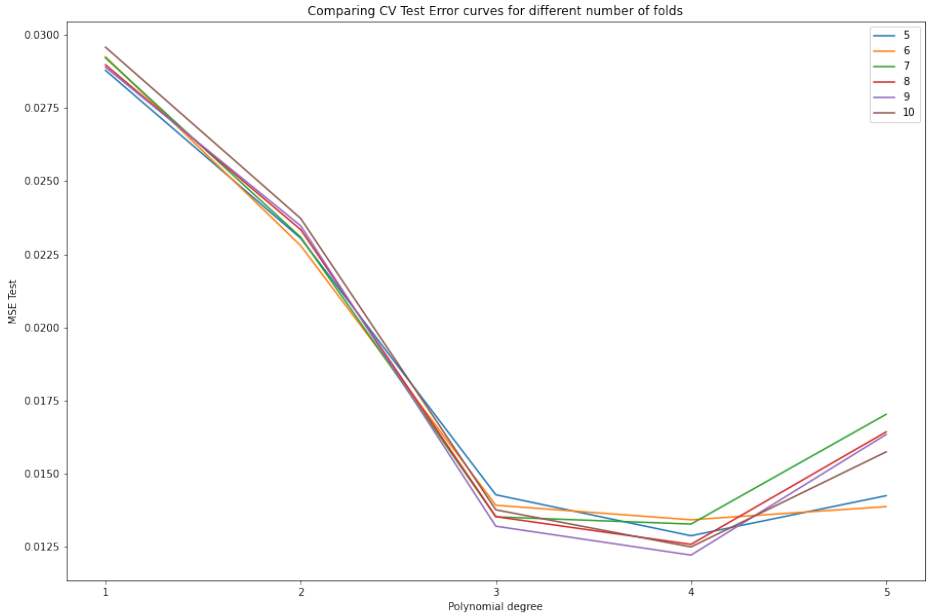

Figure 15: Comparing error from different number of folds.

3.4 Ridge regression analysis

We now apply Ridge regression on the generated data, in order to compare its performance in respect of the OLS one.

3.4.1 Ridge coefficients as function of λ

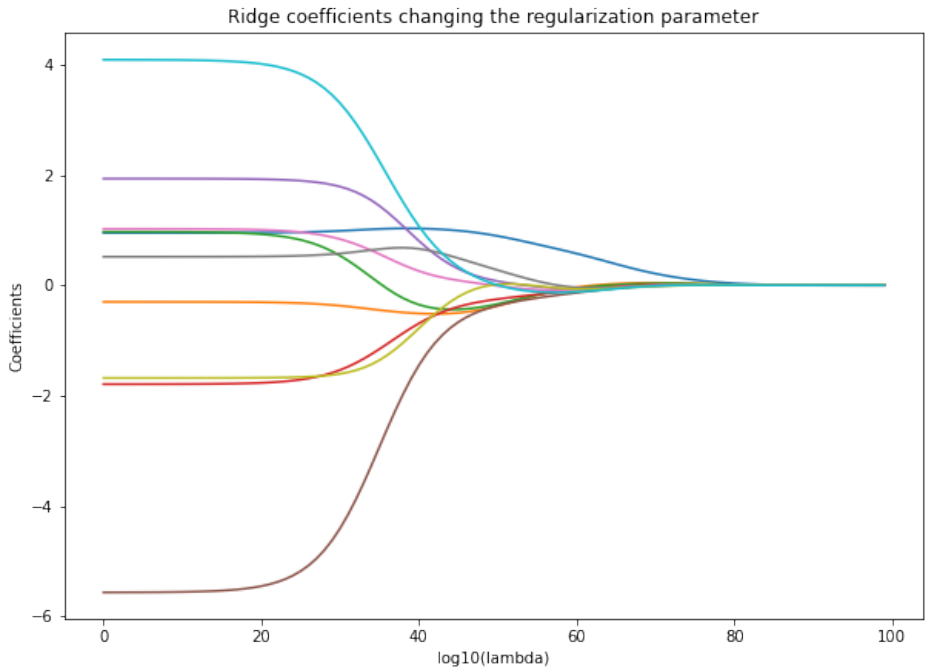

Figure 16: Ridge coefficients as a function of the regularization parameter

To obtain this graph we considered a polynomial of degree 3, which contains 10 terms. The graph shows how the coefficients vary with respect to an increasing value of λ. We notice that the parameters are gradually shrunk towards 0.

3.4.2 Bias-variance trade-off with different λ for ridge regression

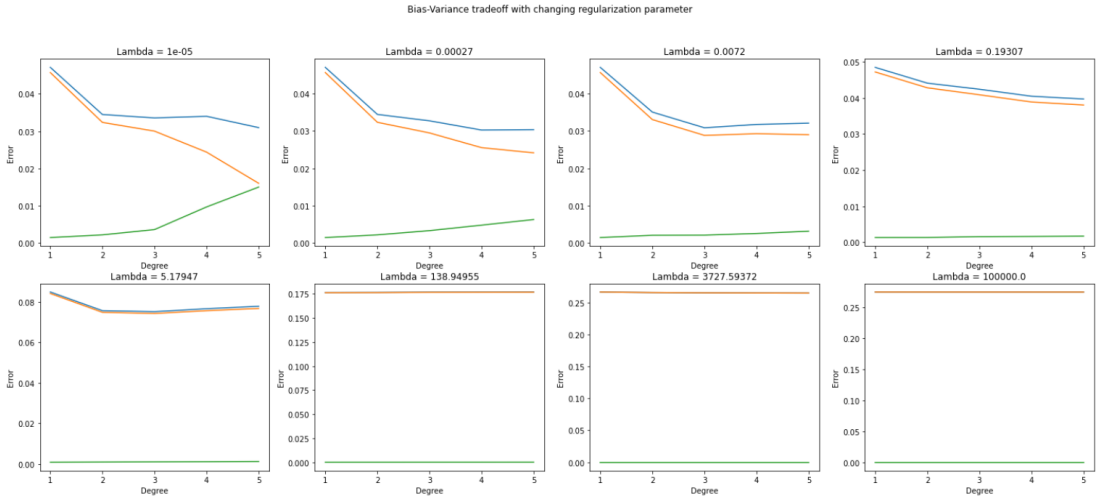

Figure 17: Bias-variance trade-off with changing regularization parameter
Note: *Blue line refers to MSE, Orange line to the bias, Green line to variance*

Figure 16 shows the variation of the bias-variance trade-off with changing regularization parameter.
While we increase the value of λ the variance decreases going to 0, because the coefficients are shrinking. On the other hand, higher values of λ introduce a stronger constraint on the model, increasing the bias. Hence, the choice of the parameter λ is crucial: the implementation of shrinkage methods can be helpful when dealing with data with high variance, but an excessively high regularization parameter can lead to an increased MSE.

3.5 Lasso Regression analysis

We now extend our analysis to lasso regression. To apply this method to our data, we introduced the **Lasso** function provided in the **sklearn.linear_model** package.

3.5.1 Lasso coefficients as function of λ

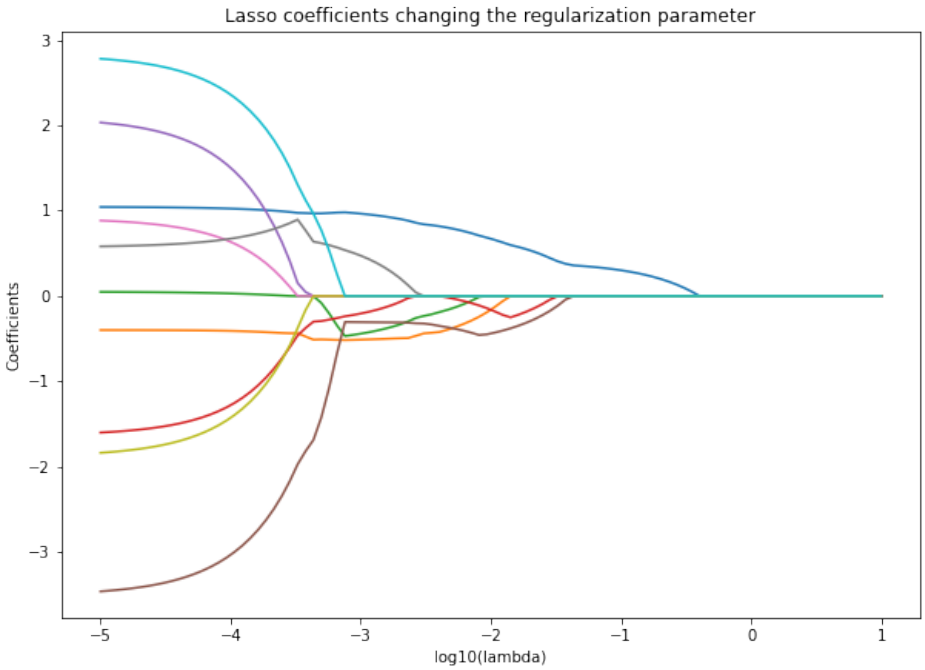

Figure 18: Lasso coefficients changing the regularization parameter

Differently from what we saw in the ridge analysis, as λ increases, the values of β do not decrease gradually, but are rather sequentially set to 0.

3.5.2 Bias-variance trade off with different λ for Lasso regression with bootstrapping

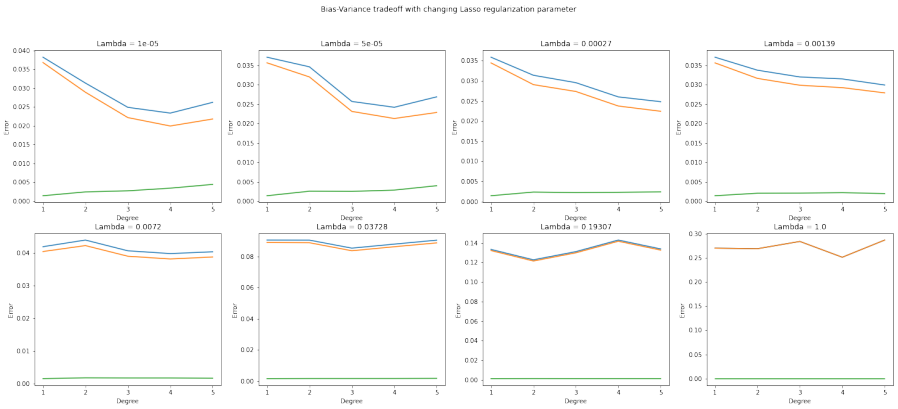

Figure 19: Bias-Variance tradeoff with changing Lasso regularization parameter
Note: *Blue line refers to MSE, Orange line to the bias, Green line to variance*

We now analyse the variance bias trade-off as a function of the polynomial degree for increasing λ.
In the case of Lasso, we have explored a smaller interval in the values of λ (from 10^{-5} to 1): for values of the parameter higher than this all coefficients get shrunk to zero and the resulting model is null. We notice that as in ridge regression the variance goes to zero while the bias is getting higher.

3.6 Comparison

After considering the three regression methods we have mentioned, we are now able to perform a comparison of their performance in terms of the test MSE. The following graphs are obtained by considering a polynomial of degree three, since it was the best performing one during the OLS analysis.

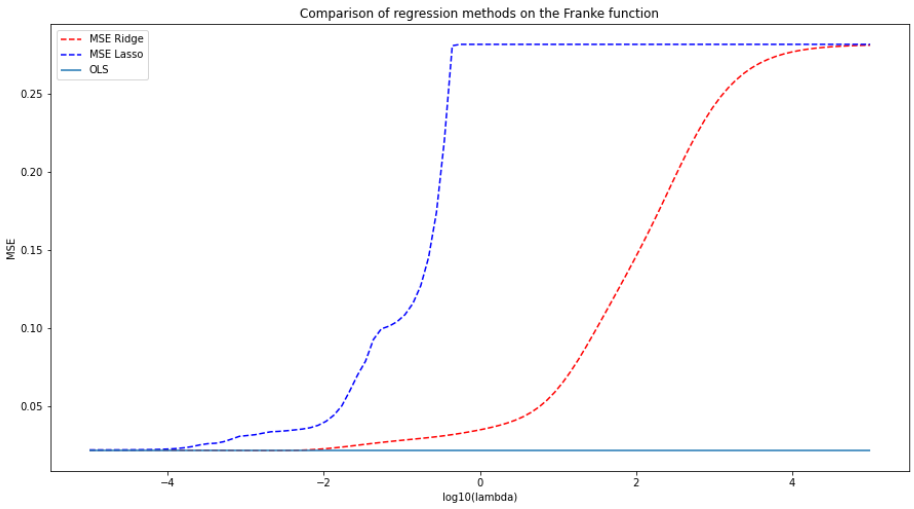

Figure 20: Comparison of regression methods on the Franke function

As shown in Figure 19, Ridge and Lasso are not able to outperform regular OLS on the fitting of our data with third degree polynomials: the variability in our data is not substantial enough to justify the use of a shrinkage method over OLS. As expected, with values of λ approaching 0, Ridge and Lasso are reduced to regular linear regression. When the value of λ increases over 10^{-1}, the MSE for Lasso reaches a plateau, meaning that all the coefficients have been shrunk to 0 and the resulting model is null. The same MSE value is more slowly approached by Ridge regression when the regularization parameter increases.

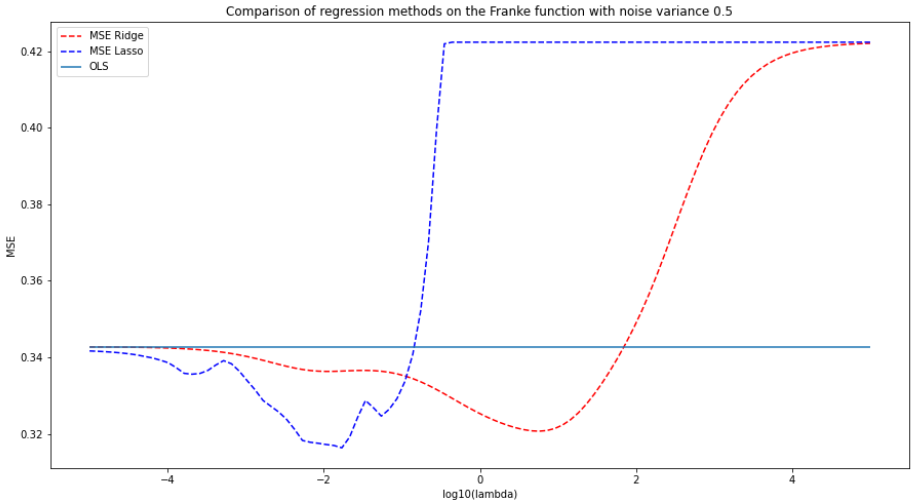

Figure 21: Comparison of regression methods on the Franke function with noise variance 0.5

In effort to deepen our understanding of the situations in which the use of shrinkage methods is beneficial over regular OLS, we considered the same framework and polynomial degree with an increased noise variance value. In this case, the analysis shows that shrinkage methods are able to achieve lower MSE values by effectively reducing the excessive variability in the data.

3.7 Analysis of real data

Having tested our methods on artificially generated data, we move to studying a real digital terrain dataset[5].
The data represents the graphical coordinates close to Stavanger in Norway and is depicted in the following figures:

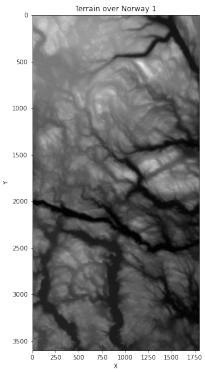

Figure 22: Terrain 2D plot

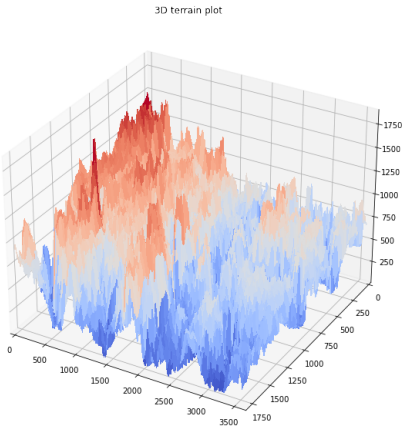

Figure 23: Terrain scatterplot

The original data consists of a grid of values of dimension 3600x1800.
To avoid the technical difficulties coming from working with such a high number of datapoints, we decided to consider 1000 random points from the map as our working dataset.

Unlike what we did for the points from the Franke function, the coordinates of the points and the z value for the real terrain data are scaled through a min-max scaling [2.5], in order to obtain values between zero and one. It is especially necessary to apply this normalization to our data, in this case, because the dimensions we are working with don't vary in the same range. This heavily influences the way shrinkage methods like Ridge and Lasso regression work, since they're not scalable: the regularization consists of introducing a penalty that is based on the magnitude of the coefficients, which is not scale invariant.

The set of points we obtained is represented in the following scatterplot:

[5]Obtained from the website `https://earthexplorer.usgs.gov/`

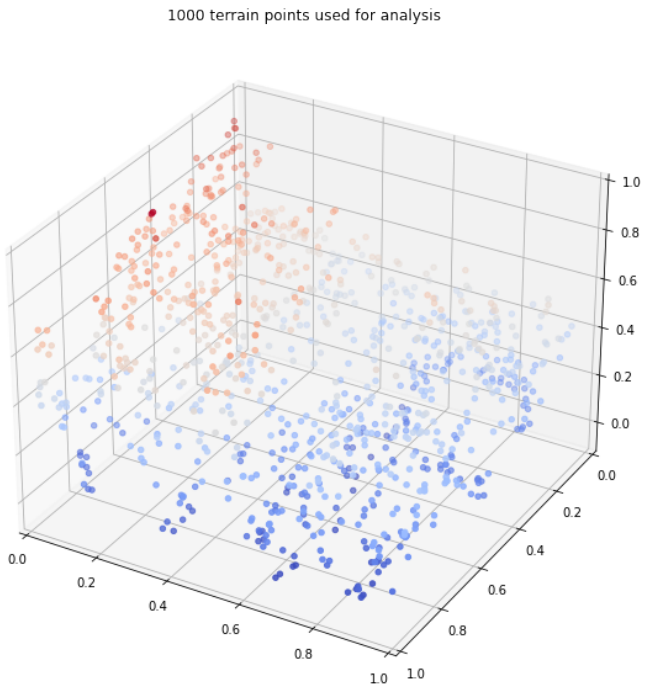

Figure 24: 1000 terrain points used for analysis

We start by analysing the results of the OLS regression with polynomials of increasing degree. The following figure depicts the values of training and test MSE obtained.

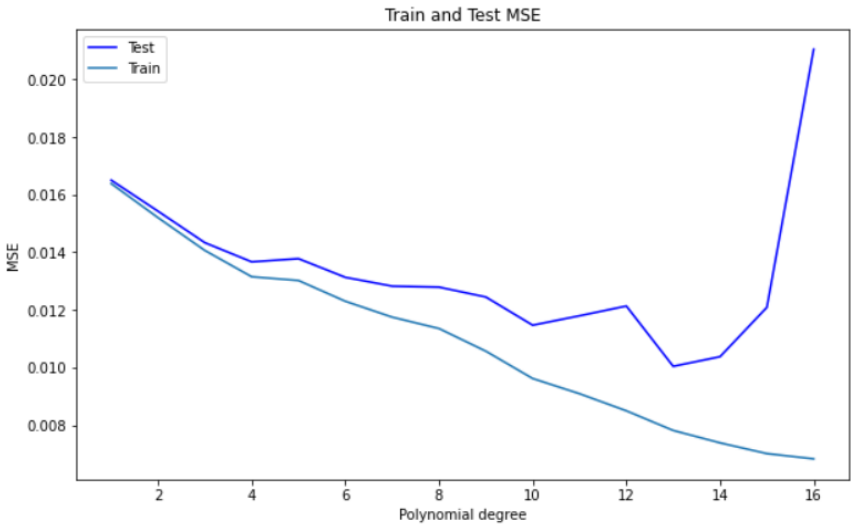

Figure 25: Test and Train MSE of Terrain data 1000 points

As we could expect, increasing the polynomial degree decreases the training error indefinitely, since the model can better adapt to the data it is trained on.
From degree 13 on, though, we can see that the test error is negatively impacted by increasing the degree: this observation tells us that the model is overfitting on the training data, and thus presents a worse performance on unseen observations.

This phenomenon can be better understood by analyzing the bias-variance decomposition of the test MSE, depicted in the following figure.

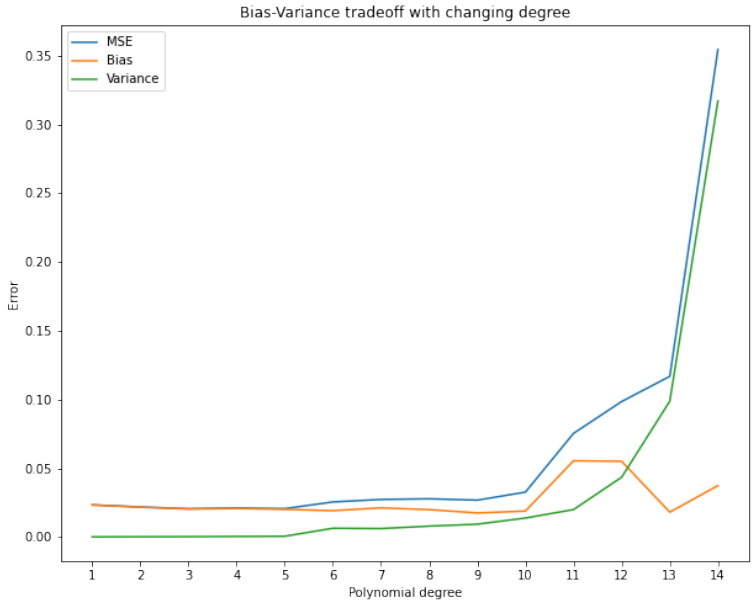

Figure 26: Bias-variance trade-off with changing degree for terrain data

This graph, obtained by implementing the bootstrap method, shows how, for increasing degrees of the polynomials used, the variance of the OLS coefficients is rapidly increased: the model is now adapting too closely to the training set and that small changes in the data are able to induce significant changes in the final model.
Differently from what we could have expected, the bias is not significantly impacted by the polynomial degree used: this unexplained pattern could be object of further investigation.

Finally, we present a comparison between the performance of the three regression models considered on the real terrain data.
The following figure depicts the test MSE of the three methods for varying values of the regularization parameter λ by using a polynomial degree of 13.

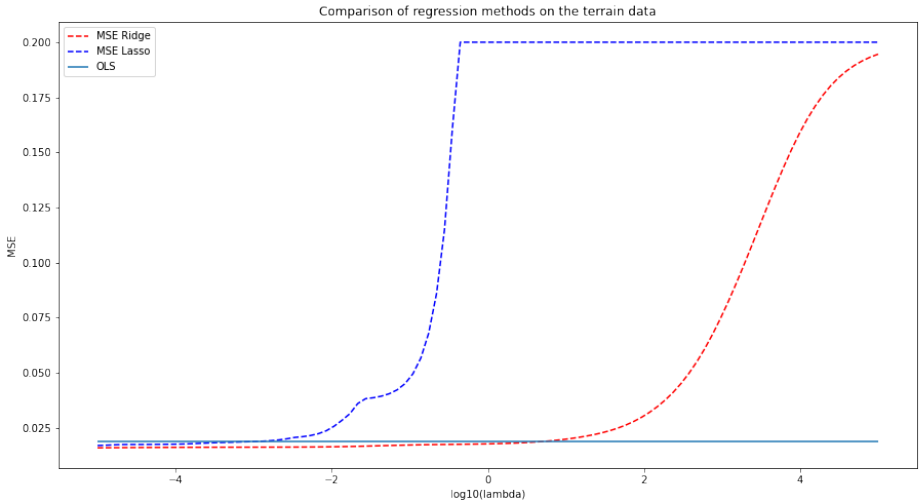

Figure 27: Comparison of regression methods on the terrain data

Once again, the values of the MSE are very comparable for all three methods for specific values of λ. From what the graph shows, we could infer that Ridge regression with values of λ around 10^{-3} could be the model with the least error out of the ones presented: however, the difference in performance is very slight, and could be influenced by the MSE estimation obtained through 10-fold cross validation. Again, the variability in the terrain data is not substantial enough to determine a significantly better performance of the shrinkage methods over the classic linear regression.

4 Conclusions and possible improvements

The aim of this report was to estimate the most suitable model comparing the performances of different linear regression implementations on two types of data. In the first case, we considered artificially generated data relative to the Franke function: the final results highlighted how, in the context of our framework, the OLS regression performed the best. This result could be due to the low variance of the generated data, which doesn't call for the use of variance-reducing regularization schemes.
The same conclusions could also be reached when considering the real terrain data: even in this case the resulting MSE for the different models shown that the OLS regression is the best choice.

As to possible improvements on our analysis, we suggest that a possible expansion could be comparing the performance of the three methods with higher polynomial degrees: in this case, while the OLS performance should be worsened due to overfitting, the use of shrinkage methods, with specific values of the parameter λ, could improve the variance term, possibly determining a total lower MSE.

References

Textbooks

[1] Hastie T., Tibshirani R., Firedman J., Springer, New York (USA), 2017, *The Elements of Statistical Learning* (Chapter 3, paragraphs 3.2, 3.4,chapter 7 paragraphs 7.2, 7.3, 7.10, 7.11)

[2] James G., Witten D., Hastie T., Tibshirani R., Springer, New York, USA, 2021, *An Introduction to Statistical Learning* (Chapter 6, 2013)

[3] Brownlee J., *Data preparation for machine learning* Machine Learning mastery: Data Cleaning, Feature Selection and Data Transforms in Python, 2020, p.217

[4] Hiep Phung V., Joo Rhee E., *A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets*, 2019, p.7

Online sources

[5] Hjorth-Jensen M., 2022, GitHub, `http://precog.iiitd.edu.in/people/anupama`, 6 October 2022

[6] Brownlee J., Machine Learning Mastery, `https://machinelearningmastery.com/k-fold-cross-validation`, *A gentle introduction to the k-fold cross validation*, 23 May 2018

[7] Brownlee J., Machine Learning Mastery, `https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/`, *A gentle introduction to the Bootstrap Method*, 25 May 2018