Regression Analysis on Franke's function and digital terrain data

Fábio Rodrigues Pereira
fabior@uio.no - github: @fabiorod

October 2020

Abstract

This project will study engaging linear regression analysis models, specifically Ordinary least squares, Ridge and Lasso. Explanatory random variables transformed in polynomials with different degrees will feed those models. Likewise, two different data types will be performed as response variables, one created by the Franke's function holding heights of a surface, and another by a GeoTIF picture containing the altitudes of a region near Stavanger in Norway. The models aim to learn from the input explanatory variables and predict the output response heights based on complexities, for example, the number of samples or polynomial degrees. We will also practice model selection methods to estimate these predictions' accuracies and choose the best model. For that, the mean square error and the coefficient of determination (R2-score) will play an essential role in assessing these predictions. Furthermore, machine learning pre-processing techniques, such as splitting, scaling, bootstrapping, and k-folds cross-validation, will be applied on different occasions for avoiding outliers, under and over-fittings. As long as the complexities increase, the models tend to over-fit, harming the performance or giving an over-estimated value. The analysis of the regularization present in Ridge and Lasso models will contribute to defeating this issue. Besides, the decomposition of the bias-variance trade-offs with bootstrapping will be crucial to understand the metrics' behavior under an increasing complexity of the data. Therefore, it will end up in an approximated view over a possible under- or over-estimation, and the conclusion will contribute to improving our forecasts by picking the most suitable model with the most desirable pre-processing methods.

Contents

1 Introduction

A sub-field of artificial intelligence (AI) called Machine Learning (ML) plays an essential role in uncountable fields nowadays. Its goals are generally to understand structured data that fits a model and predicts useful outcomes for humanity. Although machine learning is a computer science field, it has different approaches as regular programming. Typically, traditional programming only gives explicit instructions to a computer for calculating or solving problems. Machine Learning algorithms go beyond; they allow a computer to train and learn from input data and use statistical analysis to predict outputs. Due to that, computers now build models from sample data to automate decision-making processes.

Nevertheless, these ML models need to be studied and perfected by professionals and researchers. Our job here is to develop, analyze, and investigate different generalized techniques applied in as many different real domains as imaginable, with as much accuracy as possible. This duty is the motivation of our scientific report, which will explore the subject of Linear Regression analysis and see how this method can be enhanced by assessing more reliable results.

For that, this report will first investigate a vanilla data-set so-called Franke's function [3.1.1], and after a GeoTIF image of an area near Stavanger in Norway [3.6]. This Franke's function and GeoTIF image data are further denoted as z response variables and are known to generate surface heights, which polynomials can explain. Then, we generate two different sets of random explanatory variables, x and y, that will be subject to different degrees' polynomial transformation [3.1.1]. Accordingly, these sets will feed three different linear regression models, Ordinary least square, Ridge, and Lasso regression. We will examine the prediction accuracy results of their heights as a function of the explanatory polynomials. Next, these regression models will be adjusted according to each peculiarities, such as their regularization, step by step in the discussion section. The study will also perform 100x bootstrapping to visualize the bias-variance trade-off and its under or over-fitting zones. Finally, it will be essential to apply the k-folds cross-validation technique and reflect on potential improvement assessments.

It is important to say that almost all algorithms utilized in this report are self-made and available for inspection on our GitHub repository here. For that, it was created a package directory with a collection of python's code, as follows:

– accuracies.py: Containing the metrics utilized in the study, as mean-squared-error and r2-score.

– create_dataset.py: Containing the classes utilized to create the explanatory and response variables, and the design matrix.

– linear_models.py: Containing the linear regression models, for instance, OLS, Ridge, and Lasso.

– studies.py: Containing the studies performed, for example, GridSearch, Bias-variance trade-off, and k-folds cross-validation.

The following topics of this dissertation will cover these methods under the linear regression models. First, this report will approach the theories [2] utilized in the discussion section [3]. After, it will divide the discussion and result topics into different subsections, each one dealing with a different subject and with a separate python test file in our GitHub repository, such that:

– Part A [3.1]: File partA.py. Initial experiments of the design matrix, Franke's function, OLS regression, the confidence interval for the coefficients of the regression and metrics;

– Part B [3.2]: File partB.py. GridSearch and Bias-variance trade-off with bootstrapping under Franke's function and OLS model;

– Part C [3.3]: File partC.py. K-folds cross-validation under Franke's function and OLS model;

– Part D [3.4]: File partD.py. GridSearch, bias-variance trade-off with bootstrapping, and k-folds cross-validation under Franke's function and Ridge model;

– Part E [3.5]: File partE.py. GridSearch, bias-variance trade-off with bootstrapping, and k-folds cross-validation under Franke's function and Lasso model;

– Part F and G [3.6]: File partFandG.py. GridSearch, bias-variance trade-off, and k-folds cross-validation for GeoTIF image data under OLS, Ridge, and Lasso models. Additionally, the analysis of Gridsearch, bias-variance trade-off with bootstrapping, and k-folds cross-validation for each model.

In the end, a conclusion [4] will discuss the pros and cons of the methods and possible improvements and perspectives for future work.

2 Theory

2.1 Linear Regression Theory

Linear regression analysis performs a significant purpose in statistical modelling and is broadly applied nowadays. This technique is a linear approach to modelling the relationship between a response or dependent variable and explanatory or independent variables. In the case of only one explanatory variable, this is called a single linear regression, and for more than 2, we have a denominated multiple linear regression.

These relationships between variables are modelled using linear predictor functions, in other words, combinations of a set of unknown coefficients and known explanatory variables resulting in the response variables. Once we have a bunch of data observations, this is easy to infer the unknown coefficients from the techniques, and then we will end up with a model which can predict a response from any explanatory data.

Consider that we have a data-set that shows the values of an observation $\mathbf{z} = [z_0, z_1, .., z_{n-1}]$, the dependent response from a set of explanatory values $\mathbf{x} = [x_0, x_1, .., x_{n-1}]$. Since we do not know how to explain $\mathbf{z}$ in terms of $\mathbf{x}$, in other words, we do not have any known function $\mathbf{f}(*)$ underlying this case, but there is somehow a correlation between $\mathbf{z}$ and $\mathbf{x}$, and no multicollinearity among the values of x_i, then a linear regression can be formed.

From these assumptions, it is usual to understand that $\mathbf{z}$ and $\mathbf{x}$ have a linear relationship between each other, and we can create a parametrization of the unknown function $\mathbf{f}(*)$ from, for instance, a polynomial of degree (p), such that:

$$z = z(x) \to z(x_i) = z_i = \tilde{z}_i + \epsilon_i = \sum_{j=0}^{p} \beta_j x_i^j + \epsilon_i \tag{1}$$

For that, we needed to introduce a coefficient β_i and an independently and normally distributed noise ϵ_i, which are the foundation of the linear regression model, hence these raises the set of the following equations:

$$\begin{aligned}
z_0 &= \beta_0 + \beta_1 x_0^1 + \beta_2 x_0^2 + \cdots + \beta_{n-1} x_0^p + \epsilon_0 \\
z_1 &= \beta_0 + \beta_1 x_1^1 + \beta_2 x_1^2 + \cdots + \beta_{n-1} x_1^p + \epsilon_1 \\
z_2 &= \beta_0 + \beta_1 x_2^1 + \beta_2 x_2^2 + \cdots + \beta_{n-1} x_2^p + \epsilon_2 \\
&\vdots \\
z_{n-1} &= \beta_0 + \beta_1 x_{n-1}^1 + \beta_2 x_{n-1}^2 + \cdots + \beta_{n-1} x_p^{n-1} + \epsilon_{n-1}
\end{aligned}$$

We now consider z_i as a stochastic values which approximates to the real z. Besides, the interpretation of z_i is the expectation of a response z_i given a explanatory observation x_i, such that: $z_i = E[z_i|x_i]$.

Then, we re-writing everything as vectors and matrices:

$$\mathbf{z} = [z_0, z_1, z_2, ..., z_{n-1}]^T$$

$$\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \dots, \beta_{p-1}]^T$$
$$\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1, \epsilon_2, \dots, \epsilon_{n-1}]^T$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_0^1 & x_0^2 & \dots & x_0^p \\ 1 & x_1^1 & x_1^2 & \dots & x_1^p \\ 1 & x_2^1 & x_2^2 & \dots & x_2^p \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_{n-1}^1 & x_{n-1}^2 & \dots & x_{n-1}^p \end{bmatrix}$$

where $\mathbf{X} \in R^{n \times p}$ is a design matrix from p quantity of explanatory variables (columns) with n number of samples (rows).

Finally, equation (1) can be written as:

$$\boldsymbol{z} = \tilde{\boldsymbol{z}} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2}$$

2.1.1 Ordinary least squares

Ordinary least of squares is a technique of estimating the unknown coefficient β from the model.

We pick a set of $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ from the minimization of the residual sum of squares: $RSS(\beta) = \sum_{i=1}^{N}(z_i - f(x_i))^2 = \sum_{i=1}^{N}(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$, where the training observations (x_i, z_i) are a representation of independent random draws from their population, or z_i's are conditionally independent given inputs x_i.

Denote $\mathbf{X}$ as the $N \times (p+1)$ matrix where each row is an input vector, and $\mathbf{z}$ is the (N-vector,) of outputs. Then, $RSS(\beta) = (\mathbf{z} - \mathbf{X}\beta)^T(\mathbf{z} - \mathbf{X}\beta)$ is a quadratic function with $(p+1)$ coefficients. Assuming that $\mathbf{X}$ has full column rank, $\mathbf{X}^T\mathbf{X}$ is positive definite and differentiating with respect to β, we obtain:

$$\hat{\beta} = E[\beta] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z} \tag{3}$$

where $\hat{\beta}$ is the optimal beta, or minimized beta.

Since we got the minimized $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}$, then the predicted values are

$$\tilde{\mathbf{z}} = E[y] = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z} \tag{4}$$

where it is explained as the projection of $\mathbf{z}$ onto the column space of $\mathbf{X}$.

2.1.2 Confidence Interval of the coefficients β

Assume that the observations z_i are uncorrelated and have constant variance σ^2, and that the x_i are fixed. Then, the variance-covariance matrix of the least square coefficient estimates is given by:

$$Var[\hat{\beta}] = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2 \tag{5}$$

Where the variance σ^2 can be estimate by $\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(z_i - \hat{z}_i)^2$. The $(N-p-1)$ makes the $\hat{\sigma}^2$ an unbiased estimate of σ^2 by $E[\hat{\sigma}^2] = \sigma^2$.

Then, the confidence interval for β is

$$(\hat{\beta}_j - Z_{1-\alpha}(\mathbf{X}^T\mathbf{X})^{-1}\sigma^2, \hat{\beta}_j + Z_{1-\alpha}(\mathbf{X}^T\mathbf{X})^{-1}\sigma^2) \tag{6}$$

2.1.3 Ridge Regression

The OLS technique has a low bias but a large variance. It also treats all parameters with the same weight, not considering that some predictors may be more important than others. Then, Ridge and Lasso take place.

To improve the accuracy of a model, we can reduce or increase some of the coefficients, so-called regularization. Ridge regression is a technique that shrinks the coefficients, some of then turning to zero, by attributing a penalty to their size.

We introduce the tuning parameter $\lambda >= 0$ that is the amount of shrinkage. Then, the cost function will be:

$$C(\boldsymbol{\beta}) = (\boldsymbol{z} - \boldsymbol{X\beta})^T(\boldsymbol{z} - \boldsymbol{X\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

where its penalty is $||\boldsymbol{\beta}||_2^2 = \boldsymbol{\beta}^T\boldsymbol{\beta}$.

From it's derivative with respect to β, we end up with:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{z} \tag{7}$$

and the outputs $\tilde{\boldsymbol{z}}$ for a given $\boldsymbol{X}$ is:

$$\tilde{\boldsymbol{z}} = \boldsymbol{X\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T \tag{8}$$

Note that $\lambda = 0$ will results to the OLS.

2.1.4 Lasso Regression

Another regression technique is called Lasso Regression, or Least Absolute Shrinkage and Selection Operator. Its cost function is:

$$C(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta) + \lambda\sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}$$

where its penalty/regularization is $||\boldsymbol{\beta}||_1 = \sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}$.

It turns out that this penalty transforms the equation into a non-linear form, and then there is no similar linear algebraic solution for β as OLS and Ridge. Instead, we use gradient descent to minimize the cost function.

2.2 Machine Learning pre-processing methods

2.2.1 Splitting data-set

Typically, a pre-processing step needs to be done before fitting and predicting the data in a Linear Regression model. Many techniques could be applied here, but we first split the data using the sci-kit learn function train_test_split(X, z, test_size=0.2, random_state=10). This function is the most basic model selection procedure where divides a single dataset of X and z into four different blocks, X_train, X_test, z_train, and z_test, for training and testing purposes. The training sets fit and build the Linear regression model, and the testing sets are for using this built model on unknown values of X_test for predicting and evaluating the accuracy performance. This model selection is necessary because we can not accurately assess the predictions' accuracy using only data that the predicting model already knows.

Therefore, first, one trains the model with the X_train and z_train sets and then test this same model using other sets X_test and z_test to generate proper measures for the predictions.

2.2.2 Scaling data-set

Another important step, which provides relevant benefits for the linear regression model's accuracy measures, is scaling. This method standardizes and transforms the values of X_train and X_test in order to avoid outliers values that can negatively impact the accuracy of the model.

The formula for standardization is: $Z = (x - u)/s$,

where: x is each element in a column of X_train[:, C], u is the mean of all elements in a column of X_train[:, C], s is the standard deviation of all elements in a column of X_train, Z is the standardized value of x.

To do that, we use another sci-kit function called StandardScaler.

2.3 Accuracy measurements

2.3.1 Mean Squared Error

The first accuracy measure called Mean Squared Error (MSE), which estimates the average of the errors' squares, is given by the following formula:

$$MSE(\hat{z}, \hat{\tilde{z}}) = \frac{1}{n} \sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2, \tag{9}$$

Since the MSE measures the average squared difference between the predicted values and the real value, it is always non-negative, and values closer to zero are desirable.

The function is present in the file accuracies.py in our package/ directory.

2.3.2 R2-score

In its turn, the coefficient of determination, or also called r-squared or r2-score, is the rate of variation of a dependent variable (z), which is explained by the independent variables (X), given by the formula:

$$R^2(\hat{z}, \tilde{\hat{z}}) = 1 - \frac{\sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1} (z_i - \bar{z})^2}, \tag{10}$$

Since r2-score is a rate, its values are up to 1, which 1 indicates the perfect score of the model's accuracy; in other words, it means that the explanatory variables explain 100 percent of the response variable.

The function is present in the file accuracies.py in our package/ directory.

2.3.3 Bias-variance trade-off

The bias-variance trade-offs play an essential role in Machine Learning when assessing the predicting model's accuracy.

Bias is the measure of the distance between our estimate from the real target value. Higher bias tends to underestimate/under-fit the data because it oversimplifies the training.

Variance measures the sensitivities of the model. Typically, it can be shown by the difference between the MSE train and the MSE test. Again from (Fig.5), the higher contrast between both metrics, the higher variance. Higher variance implicates overestimating/over-fitting the data because it over-complicates the training.

The sweet spot here is to find a model with low bias and low variance. Thus, the more complex are the factors, the lower the (squared) bias, but the higher the variance.

The drawing below summarizes the effect of bias and variance in the accuracies' prediction:

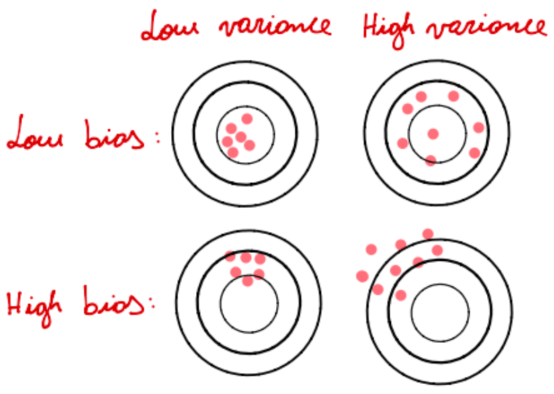


Figure 1: The effect of bias and variance in the accuracies' prediction.

From the previous explanation of the ordinary least squares method [2.1.1], assumes our model $\tilde{z} = \mathbf{X}\boldsymbol{\beta}$.

The parameters betas can be found by optimizing the means squared error via the so-called cost function:

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

From this, one can prove the bias-variance relationship:

$$\begin{aligned}
E[(\boldsymbol{z} - \tilde{\boldsymbol{z}})]^2 &= E[(f + \epsilon - \tilde{\boldsymbol{z}})]^2 \\
&= E[(f + \epsilon - \tilde{\boldsymbol{z}} + E[\tilde{\boldsymbol{z}}] - E[\tilde{\boldsymbol{z}}])^2] \\
&= E[(f - E[\tilde{\boldsymbol{z}}]^2)] + E[\epsilon^2] + E[E[(\tilde{\boldsymbol{z}} - \tilde{\boldsymbol{z}})^2] + 2E[(f - E[\tilde{\boldsymbol{z}}]\epsilon)] \\
&+ 2E[\epsilon(E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}})] + 2E[(E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}})(f - E[\tilde{\boldsymbol{z}}])] \\
&= (f - E[\tilde{\boldsymbol{z}}])^2 + E[\epsilon^2] + E[(E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}})^2] \\
&+ 2(f - E[\tilde{\boldsymbol{z}}])E[\epsilon] + 2E[\epsilon]E[E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}}] + 2E[E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}}](f - E[\tilde{\boldsymbol{z}}]) \\
&= (f - E[\tilde{\boldsymbol{z}}]^2 + E[\epsilon^2] + E[(E[\tilde{\boldsymbol{z}}] - \tilde{\boldsymbol{z}})^2] \\
&= (f - E[\tilde{\boldsymbol{z}}])^2 + \sigma^2 + Var[\tilde{\boldsymbol{z}}] \\
&= Bias(\tilde{\boldsymbol{z}})^2 + \sigma^2 + Var(\tilde{\boldsymbol{z}})
\end{aligned}$$

where,

$$\begin{aligned}
Var(\mathbf{z}) &= E[(\tilde{\boldsymbol{z}} - E[\tilde{\boldsymbol{z}}])^2] \\
&= E[(\mathbf{y} - f)^2] \\
&= E[(f + \epsilon - f)^2] \\
&= E[\epsilon^2] \\
&= Var[\epsilon] + E[\epsilon]^2 \\
&= \sigma^2
\end{aligned}$$

and,

$$\begin{aligned}
Bias(\tilde{\boldsymbol{z}}) &= E[\tilde{\boldsymbol{z}} - f] \\
&= E[\tilde{\boldsymbol{z}}] - f
\end{aligned}$$

3 Discussion with results

3.1 Part A of the study

In this chapter, we will describe, step by step, how to create algorithms for randomly generated data, pre-process the data before usage (splitting and scaling), build a linear regression model, assess its accuracy measures, and finally apply everything together.

3.1.1 Creating data-set from random sets

We initiate this Linear Regression study generating a random data-set for testing our own designed algorithms and package. Besides, these created sets will fit a known vanilla function: Franke's Function. This function will give the response variables which we are interested in predicting further later.

First, create_data.py with a "CreateData" class is created in our package directory package/ to generate the random data-sets. Also, a collection of class methods supports the formation of the data-set, especially:

_design_matrix(x, y, degree): This takes three parameters, two sets of explanatory variables, and the degree of the polynomial we plan to produce. Then, returns the polynomial transformation of that in a so-called design matrix X of shape (N, p), where N is the number of all combinations of samples/observations/rows, and p is the number of terms of the polynomial given by the degrees/-complexities/factors of the data-set.

_franke_function(x, y): This takes two parameters, two sets of explanatory variables/observations, and returns the following calculation as a column vector z of shape (N, 1):

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

_plot(x, y, z): Returns a 3d plot from the flattened sets x, y, z.

Fitting the two sets x and y, of n random numbers between 0 and 1, in the CreateData class, we will end up with the X design matrix and z column vector, all of which are necessary to perform the studies.

However, since the set x, y, z will return a well-defined surface in the 3D universe [left of Fig.1], which will not proffer any challenge for our Linear Regression models, we added 15 percent of a standard normally distributed noise to the response z. Then, one can see that a new 3D plot of the surface contains many choppy points around [right of Fig.1], denoting the noise and increasing our predictions' difficulty.

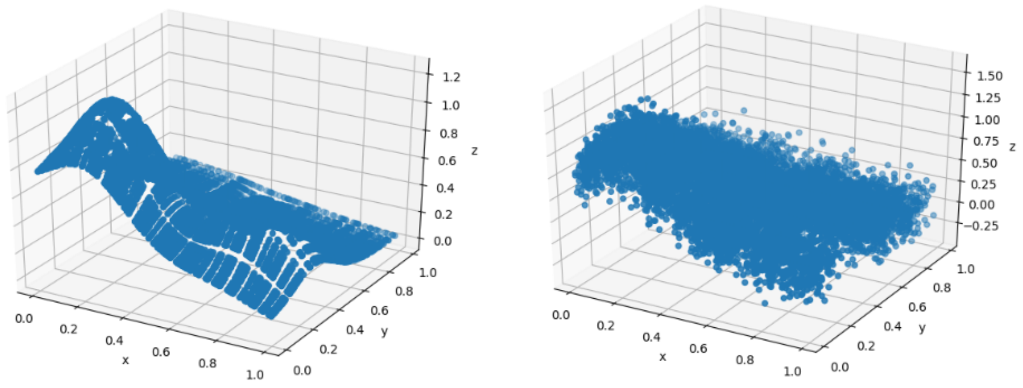


Figure 2: Surfaces generated from sets x and y of 100 random numbers in the interval 0, 1, and set z from FrankeFunction [3.1.1]

3.1.2 Fitting the OLS model and predicting the targets

From the theory consolidated in Linear Regression's theory [2.1] of this report and using the created and pre-processed data by splitting [2.2.1] and scaling [2.2.2], we now develop an ordinary least squares algorithm to predict our target z (response vector).

In the package/ directory, a file called linear_models.py contains an OLS class that performs the fitting, predictions, and confidence interval calculations.

First, initialize and fit OLS with X and z generated and pre-processed by CreateData class, train_test_split(X, z, test_size=0.2, random_state=10) and StandardScaler.

The theory, described in [2.1.1], explains how to find the expected coefficient vector beta while fitting the model. For that, the numpy linear algebra package does the computation job, such that:

```
numpy.linalg.pinv(X.T @ X) @ X.T @ z
```

After fitting OLS, the methods predict(X_train) and predict(X_test) performs the prediction for the training and testing data, respectively. Again, numpy linear algebra:

```
X @ self.coef_
```

where x is X_train or X_test, and coef_ is the vector of expected betas.

The confidence interval for the vector beta is described by the theory in [2.1.2] and coded as an OLS class method called coef_confidence_interval.

Finally, we can assess the accuracy measures after predictions.

3.1.3 Part A - Outputs and conclusion

In this 'Part A' of our study, we ran our code for different polynomial transformations of degrees 2, 3, 4, 5, and for nr_samples=12, and got as results the outputs present in the appendix [6.1]. Also, one can run the file partA.py to replicate the results.

For the training predictions, the outputs show MSE scores between 0.027 and 0.019, and R2-scores between 65 and 75 percent.

For the testing predictions, the outputs show MSE scores between 0.069 and 0.035 and R2-score between 65 and 77 percent.

Thus, the z_train and z_test predictions are near the real values. Also, the factors in the design matrix explain the target (z) at a rate of approx. 65 to 77 percent. Notice that as the polynomials' complexity increases from 2 to 5 as the r2-score increases and the MSE score decreases, it suggests that we could increase the polynomial degrees to acquire better results.

3.2 Part B of the study

From Part A, we learned:

- how to create the design matrix X and target z;

- splitting and scaling the train and test sets;

- fitting the linear regression model;

- predicting the z-train and z-test targets;

- evaluating the accuracy metrics MSE and r2-score.

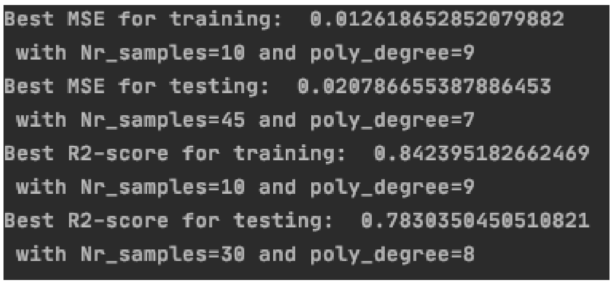
Now, we will cover more specifics topics related to bias-variance trade-off and re-sampling techniques.

3.2.1 Finding a benchmark for our study

Before continuing our research, let's analyze a little more the OLS model's accuracy and find a benchmark for our study. Later on, we will try to beat this benchmark score using other linear regressions or machine learning techniques.

First, the file studies.py located at package/ directory contains a class function called GridSearch, which takes a range of values for both nr_samples, poly_degrees, and returns vectors of MSE and R2-score from all possible combinations of these two first input parameters.

Bellow, there is a printed output of a GridSearch of nr_samples between 10 and 45, and poly_degrees between 2 and 9:



```
Best MSE for training:  0.012618652852079882
 with Nr_samples=10 and poly_degree=9
Best MSE for testing:  0.020786655387886453
 with Nr_samples=45 and poly_degree=7
Best R2-score for training:  0.842395182662469
 with Nr_samples=10 and poly_degree=9
Best R2-score for testing:  0.7830350450510821
 with Nr_samples=30 and poly_degree=8
```

Figure 3: Output of the GridSearch for nr_samples between 10 and 45, and poly_degrees between 2 and 9.

One can see here that the best/minimal MSE obtained for training sets was approx. 0.012 for nr_samples equals 10 and poly_degree 9. On the other hand, for testing sets, the best/minimal MSE was 0.020 at nr_samples 45 and poly_degree 7.

In the same output, the best/maximal r2-score for training sets was approx. 0.84 at nr_samples 10 and poly_degree 9. For testing, it was 0.78 for nr_samples 30 and poly_degree 8.

These results show an accuracy improvement from the exercise partA [6.1]. However, we need always to be cautious about under-fitting or over-fitting our model, as we can check more about it in the following heat-map plots:

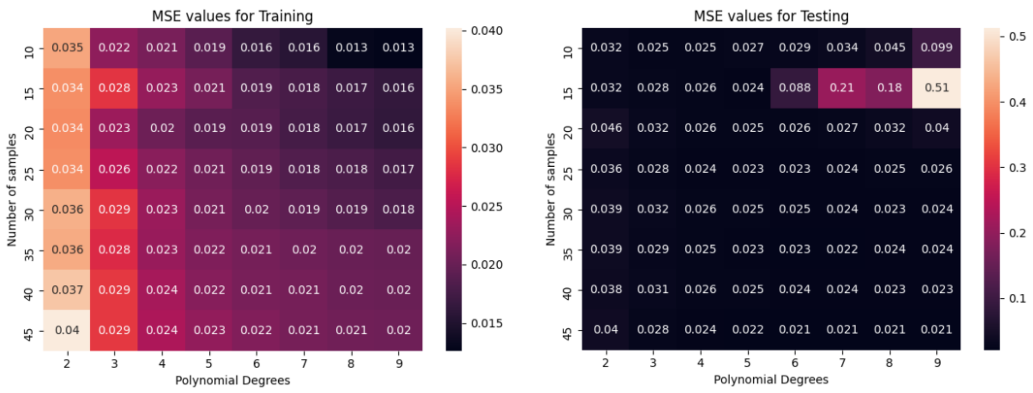


Figure 4: Heat-map containing the MSE train and test results of the GridSearch.

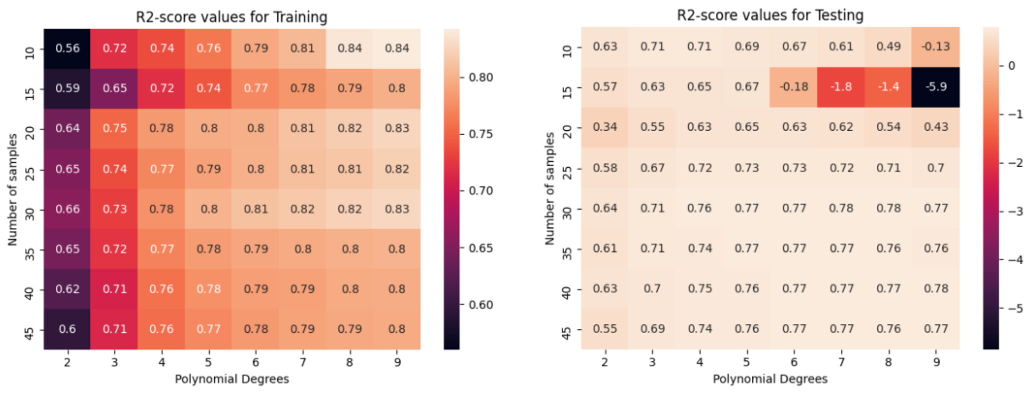


Figure 5: Heat-map containing the R2-score train and test results of the GridSearch.

From left to right, these heat-maps show that the accuracy of the predictions, for both metrics (MSE and R2-score), starts not desirable for a lower polynomial degree (under-fitting). As long as the polynomial degree increases, the metrics improve until a certain point and then worsen again (over-fitting).

The same happens when we take into consideration the top to the bottom of the heat-maps. The metrics start higher (under-fitting), and as long as the number of samples increases, the metrics improve until a certain point and then worsen again (over-fitting).

Let's analyze a similar plot as we can find in page 38 and 220 in *Hastie, Tibshirani, and Friedman*, but with the data and parameters of our case:

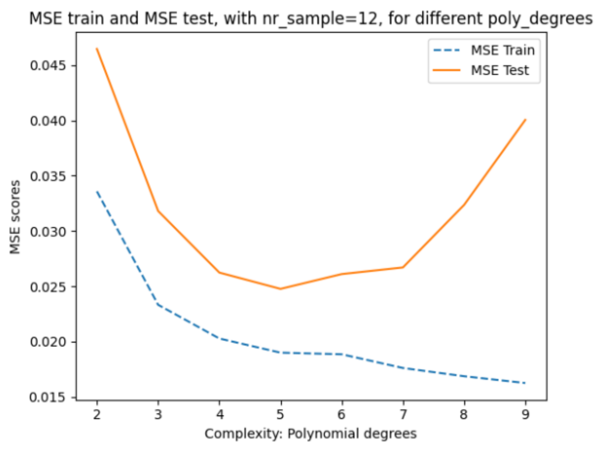


Figure 6: Line plots showing the MSE training and testing for parameters nr_samples=20 and poly_degrees from 2 to 9.

Notice that the plot's left side is a higher bias and lower variance region, the middle side is a lower bias and lower variance zone, and the right side is a higher bias and higher variance area. Hence, the sweet spot, for the complexity poly_degree given nr_sample=20, is around 4 or 5, where the testing MSE gets the best metric with low bias and variance plus closest to the training MSE.

As long as the complexity (poly_degrees) increases, the scores get better values until a certain point, where after there is an expansion of the model's variance and the metrics worsen. This enlarged variance may indicate an over-fitting, which we will discuss in detail in the next subtopic.

For didactic purposes, we choose the number of samples equals 20 because there is an perfect behavior for visualizing the bias-variance trade-off decomposition.

Therefore, we choose our parameter nr_sample equals 20 for our benchmark of this study.

3.2.2 Bias-variance decomposition with bootstrapping technique

Now, we can code an algorithm to illustrate the bias-variance trade-off decomposition as explained before in [2.3.3] in more detail. For that, we use a re-sampling technique called bootstrapping. The file studies.py in the package/ directory has a class called BiasVarianceTradeOff that returns the decomposition and a plot.

The bootstrapping is a simple technique to re-sample the arguments X_train and z_train and predict each re-sampled response.

Let us analyze the plot for nr_samples=20 and poly_degrees between 2 and 8:

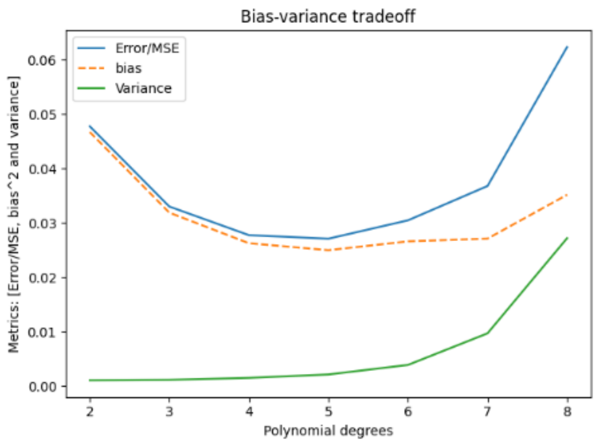


Figure 7: Bias-variance decomposition for nr_samples=20 and poly_degrees between 2 and 8.

On the left side of the plot, there is a zone of high bias and low variance. As long the complexity polynomial degrees increases, the error and bias decrease until around poly_degree=5. Then, the model's variance starts to increase, showing possible over-fitting, making the error metric sky rocket. Therefore, on the right side of the plot, there is a region of higher variance.

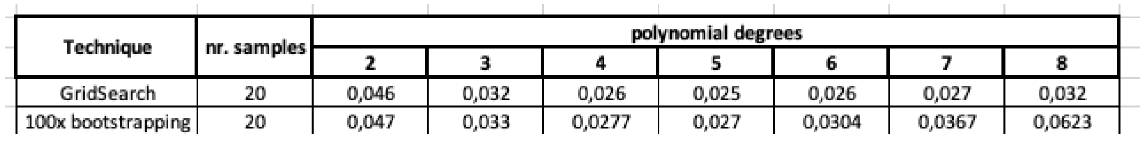
The sweet spot here, again, is when the polynomial degree is equal to 5. On this point, it is where the variance starts to increase significantly, and the error reaches the smallest score.

Thus, the poly_degree=5 will be our benchmark in the next sections.

3.2.3 Part B - Outputs and conclusion

In this part, we studied the bias-variance trade-offs and its decomposition [2.3.3 and 3.2.2]. Also, there was an implementation of a bootstrapping re-sampling model in file studies.py. The output [6.2] is in the appendix section of this report, and one can reproduce the code using the file partB.py.

The comparison between GridSearch (partA) and bias-variance trade-off with bootstrapping (partB) is below:



Technique	nr. samples	polynomial degrees						
		2	3	4	5	6	7	8
GridSearch	20	0,046	0,032	0,026	0,025	0,026	0,027	0,032
100x bootstrapping	20	0,047	0,033	0,0277	0,027	0,0304	0,0367	0,0623

Figure 8: Comparison for testing MSE scores between techniques.

We see that the bootstrapping technique has not increased the precision of the MSE scores. This unimprovement might have occurred because the model already has enough data for training, then it is not necessary to re-sampling more data. Moreover, this bootstrapping increased the model's variance for higher polynomial degrees, which might infer that this re-sampling technique increased and accelerated our model's over-fitting.

3.3 Part C of the study

This section will continue with Franke Function and Linear Regression study, but now applying a type of model selection called k-folds cross-validation.

3.3.1 K-folds cross-validation to assess model performance

To evaluate an acceptable bias-variance trade-off, we must carefully feed our model with specific training data to obtain reliable estimates on unseen samples. K-fold cross-validation is one of the well-known techniques that are widely implemented for this proposes. It splits the training data into k number of folds without replacement. One fold evaluates the predictions, and the others fit and train the machine learning model. This system is repeated k times so that we achieve k different models and estimate's scores. Then, these k scores return an estimated average performance of the model.

Typically, hyper-parameter tuning uses k-fold to assess generalization performances. The aim here is to provide more different pieces of training samples for the learning algorithm to obtain better-fitted models and more precise predictions. Also, it is an ideal method to tackle an insufficient number of samples in a data-set.

An excellent drawing, that summarizes the k-folds systems, is present on page 299 of the book *Python Machine Learning* [S. Raschka, V. Mirjalili], which is replicated here:

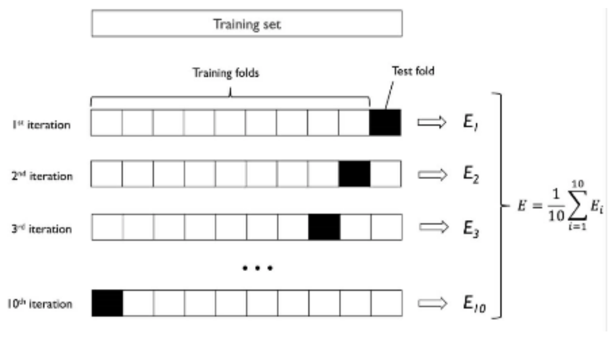


Figure 9: K-fold cross-validation system.

The file studies.py in the package/ directory has a class called CrossValidationKFolds that performs this system explained above.

Then, let's implement this technique for 5-10 folds with 20 samples and polynomial degrees 2-8:

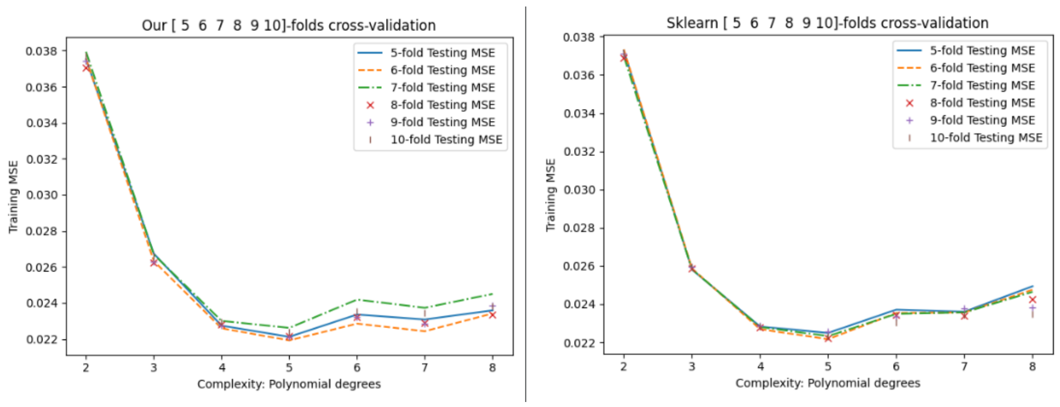


Figure 10: The left side plot is our algorithm for k-folds where k=[5, 6, 7, 8, 9, 10], nr_samples=20 and poly_degrees=[2, 3, 4, 5, 6, 7, 8]. The right side plot is a Scikit-Learn k-fold algorithm with the same parameters, but using the Linear Regression model from Sci-kit.

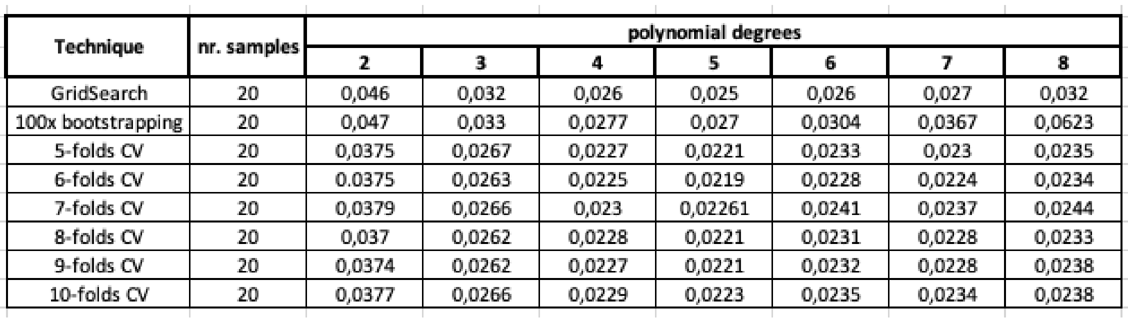
Notice that our algorithms and Sci-kit's algorithms have quite the same outputs.

The k-folds brought an attenuation in the testing MSE indications. This mitigation might have occurred because of the reduction of the model's variance. Besides, the sample space of the training data is smaller as the number of folds is higher, then it could have decreased the complexity of the model and, therefore, the variance.

3.3.2 Part C - Outputs and conclusion

In this part, we studied the k-folds Cross-Validation technique. Also, there was our implementation of k-folds cross-validation in file studies.py. The outputs are in this report's appendix section [6.3]. Also, the experiment can be reproduced by the file partC.py in our GitHub repository.

The comparison among GridSearch (partA), bias-variance trade-off with bootstrapping (partB) and k-folds cross-validation (partC) is below:



Technique	nr. samples	polynomial degrees						
		2	3	4	5	6	7	8
GridSearch	20	0,046	0,032	0,026	0,025	0,026	0,027	0,032
100x bootstrapping	20	0,047	0,033	0,0277	0,027	0,0304	0,0367	0,0623
5-folds CV	20	0,0375	0,0267	0,0227	0,0221	0,0233	0,023	0,0235
6-folds CV	20	0.0375	0,0263	0,0225	0,0219	0,0228	0,0224	0,0234
7-folds CV	20	0,0379	0,0266	0,023	0,02261	0,0241	0,0237	0,0244
8-folds CV	20	0,037	0,0262	0,0228	0,0221	0,0231	0,0228	0,0233
9-folds CV	20	0,0374	0,0262	0,0227	0,0221	0,0232	0,0228	0,0238
10-folds CV	20	0,0377	0,0266	0,0229	0,0223	0,0235	0,0234	0,0238

Figure 11: Comparison for testing MSE scores between techniques.

We see that the k-folds cross-validation technique has improved the testing MSE scores. This improvement might have occurred because of the sample space of the training data, which is smaller as the number of folds is higher, then it might have decreased the complexity of the model and, therefore, the variance.

3.4 Part D of the study

This section will continue with Franke's function [3.1.1], but now applying a different linear regression model called Ridge Regression [2.1.3]. We will also optimize the Ridge's hyper-parameter lambda, performing bootstrapping and k-folds cross-validation. In the end, it will be checked if there is any improvement in the testing MSE scores while using its shrinkage regularization.

3.4.1 Ridge's hyper-parameter lambda - Regularization

As studied in the section about linear regression theory [2.1.3], Ridge Regression aims to attenuate the OLS model's variance by introducing a regularization technique. This new parameter, $\lambda >= 0$, is added at the end of the OLS's cost function, enabling tuning the optimum betas and variance.

The heat-map below illustrates the higher lambda, the lower coefficients of the linear regression (betas) are:

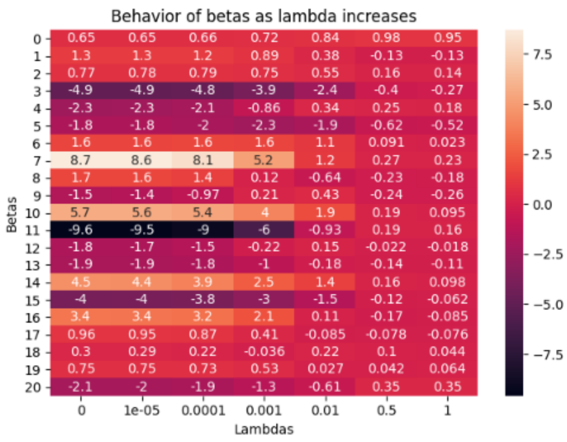


Figure 12: Heat-map showing the behavior of betas as lambda increases for nr_samples=20 and poly_degree=5.

As said before, this reduction of the betas attenuates the variance by diminishing the over-fitting. On the other hand, we have to be careful when using the lambda because very high lambda under-fits the model. This under-fit occurs because of reducing the betas, which might bring lower complexity for the model.

3.4.2 GridSearch for finding the best Ridge's lambda

Now, it is time to find the best lambda value for our data-sets with sample space of 20 (nr_samples parameter). For that, a GridSearch is run:

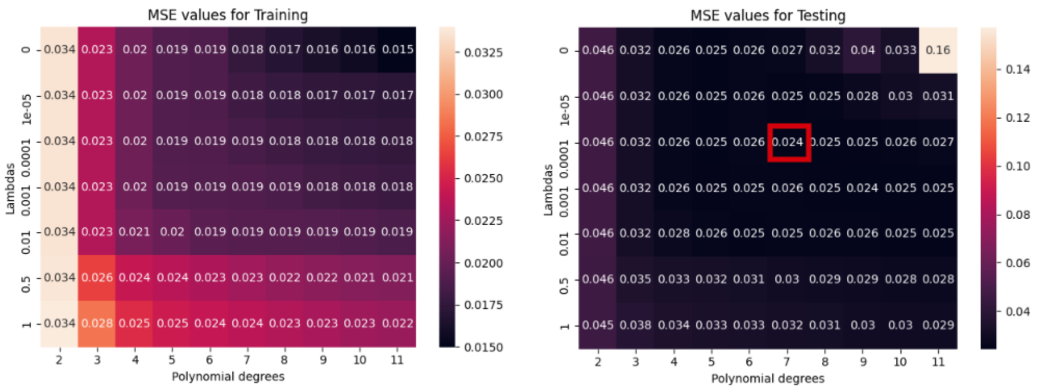


Figure 13: Heat-maps showing the behavior of training (left) and testing (right) MSE scores as lambda and polynomial degrees increase.

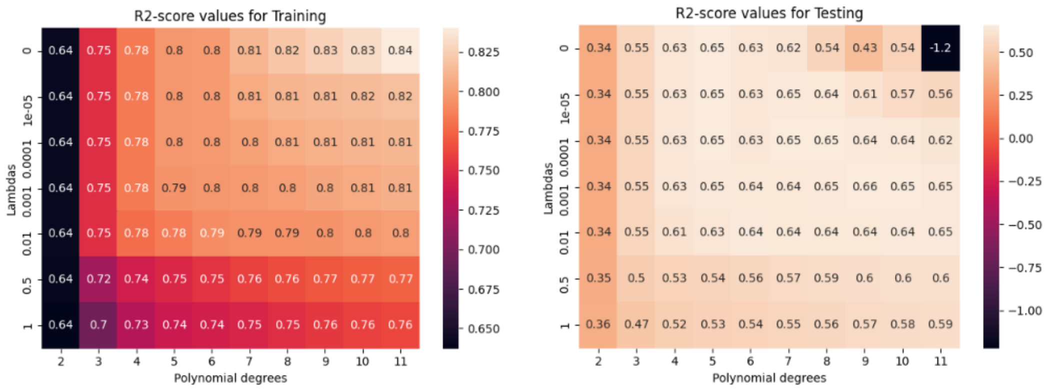


Figure 14: Heat-maps showing the behavior of training (left) and testing (right) R2-scores as lambdas and polynomial degrees increase.

Here, we are interested in finding the best MSE testing score for as lower complexity as possible, where it is seen at lambda equals 0,0001, and polynomial degree equals 7.

Notice, also, that where $\lambda = 0$ means that the Ridge model is equal to the OLS model. Thus, as expected, as long as the lambda increases, the variance for higher polynomial degrees decreases compared to the OSL model.

Therefore, lambda equals to 0,0001 is chosen for studying the training and testing MSE plot below:

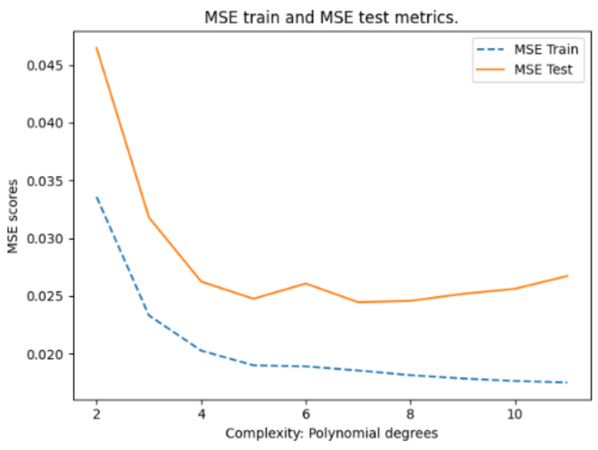


Figure 15: Line plot showing the behavior of training and testing MSE-scores for lambda=0.0001, number of samples equals 20 and polynomial degrees between 2, 12.

As expected, there is a lower variance for a higher polynomial degree than the OLS's plot [3.2.1], with the same parameters.

3.4.3 Ridge's Bias-variance trade-off with bootstrapping

Here, it was performed a 100x bootstrapping for studying the bias-variance trade-off. For that, we set lambda=0.0001, nr_samples=20, and poly_degrees from 2 to 12:

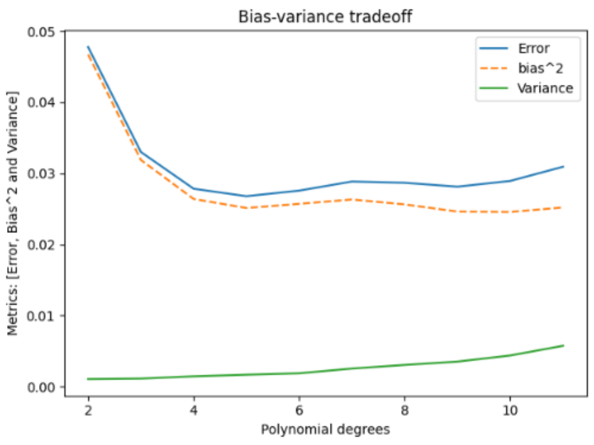


Figure 16: Line plot showing the bias-variance trade-off.

As expected again, there is a lower variance for a higher polynomial degree than the OLS's plot [3.2.2], with the same parameters.

3.4.4 Ridge's k-folds cross-validation

We continue with 20 for the number of samples and 0,0001 for lambda and, finally, run the k-folds cross-validation for polynomial degrees between 2 to 12:

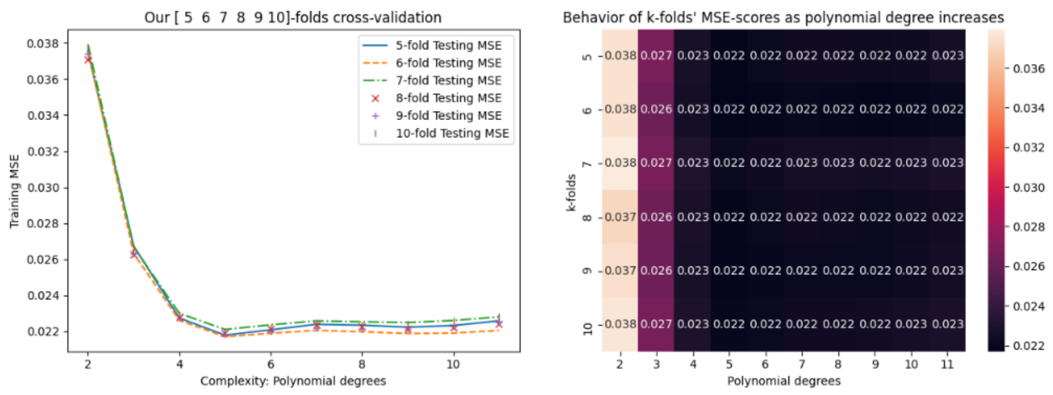


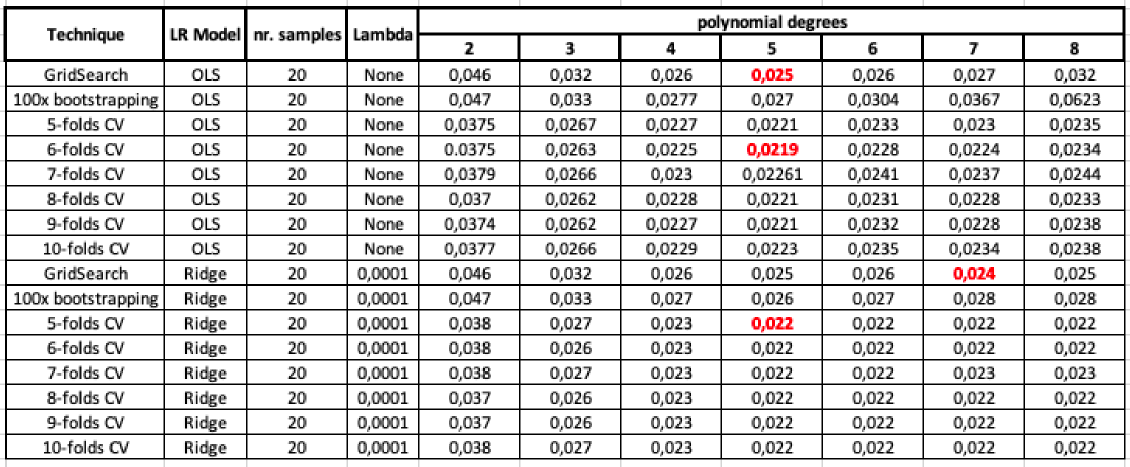
Figure 17: Left side containing line plots for different k-folds. Right side containing heat-map with the k-folds' MSE results.

Again, one can observe that the model's variance takes longer to appear, and the plots have more stable lines.

3.4.5 Part D - Outputs and conclusions

In this part, we studied the GridSearch, bias-variance trade-offs, and k-folds cross-validation techniques for Ridge regression. The outputs are in this report's appendix section [6.4]. The experiment can be reproduced by using the file partD.py in our GitHub repository.

The comparison among GridSearch (partA), bias-variance trade-off with bootstrapping (partB) and k-folds cross-validation (partC) is below:



Technique	LR Model	nr. samples	Lambda	polynomial degrees						
				2	3	4	5	6	7	8
GridSearch	OLS	20	None	0,046	0,032	0,026	**0,025**	0,026	0,027	0,032
100x bootstrapping	OLS	20	None	0,047	0,033	0,0277	0,027	0,0304	0,0367	0,0623
5-folds CV	OLS	20	None	0,0375	0,0267	0,0227	0,0221	0,0233	0,023	0,0235
6-folds CV	OLS	20	None	0.0375	0,0263	0,0225	**0,0219**	0,0228	0,0224	0,0234
7-folds CV	OLS	20	None	0,0379	0,0266	0,023	0,02261	0,0241	0,0237	0,0244
8-folds CV	OLS	20	None	0,037	0,0262	0,0228	0,0221	0,0231	0,0228	0,0233
9-folds CV	OLS	20	None	0,0374	0,0262	0,0227	0,0221	0,0232	0,0228	0,0238
10-folds CV	OLS	20	None	0,0377	0,0266	0,0229	0,0223	0,0235	0,0234	0,0238
GridSearch	Ridge	20	0,0001	0,046	0,032	0,026	0,025	0,026	**0,024**	0,025
100x bootstrapping	Ridge	20	0,0001	0,047	0,033	0,027	0,026	0,027	0,028	0,028
5-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	**0,022**	0,022	0,022	0,022
6-folds CV	Ridge	20	0,0001	0,038	0,026	0,023	0,022	0,022	0,022	0,022
7-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	0,022	0,022	0,023	0,023
8-folds CV	Ridge	20	0,0001	0,037	0,026	0,023	0,022	0,022	0,022	0,022
9-folds CV	Ridge	20	0,0001	0,037	0,026	0,023	0,022	0,022	0,022	0,022
10-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	0,022	0,022	0,022	0,022

Figure 18: Comparison for testing MSE scores between techniques.

The best testing MSE achieved with Ridge was 0,022 for 5-folds cross-validation. On the other hand, the best score for testing MSE with the OLS model was 0,0219 for 6-folds cross-validations.

Even though the Ridge regression reduced the predictions' variance, it did not reach a better score than the OLS model for polynomial degrees between 2-8.

Ridge regression makes it possible to increase the polynomial degrees (complexity of the model) with lower variance than OLS; then, it might be beneficial and might get better scores if a higher degree is applied.

3.5 Part E of the study

This section will apply a new linear regression model called Lasso. We will also optimize the Lasso's hyper-parameter lambda, performing bootstrapping and k-folds cross-validation. In the end, it will be checked if there is any improvement in the testing MSE scores while using its regularization.

3.5.1 Lasso's hyper-parameter lambda - Regularization

As studied in the section about Lasso's theory [2.1.4], the aim here is to attenuate the OLS model's variance by introducing a regularization technique. This regularization penalizes the regression's betas, the ones with less relevance for the predictions, forcing them to converge to its minimum.

As we did for Ridge's model, but now for Lasso's, the heat-map below illustrates the higher lambda, the lower coefficients of the linear regression (betas) are:

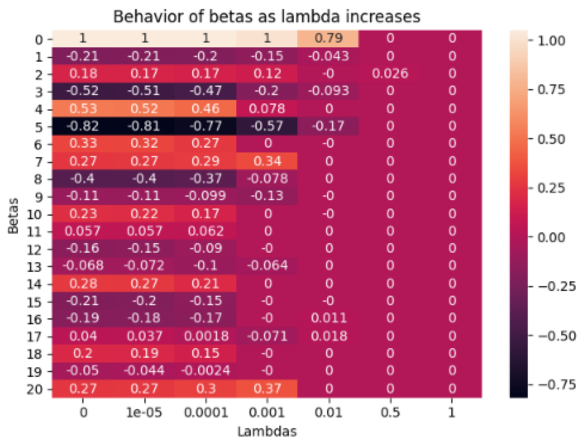


Figure 19: Heat-map showing the behavior of betas as lambda increases for nr_samples=20 and poly_degree=5

Notice that when lambda is 0, the Lasso model has the same coefficients as the OLS's model. Also, we have to be careful when using the lambda because very high lambda under-fits the model. From the heat-map, lambda equals one results in all coefficients 0, which ends up in an under-fitted model with very low complexity.

3.5.2 Finding the best lambda with GridSearch

A GridSearch finds the best lambda's value for our data-sets:

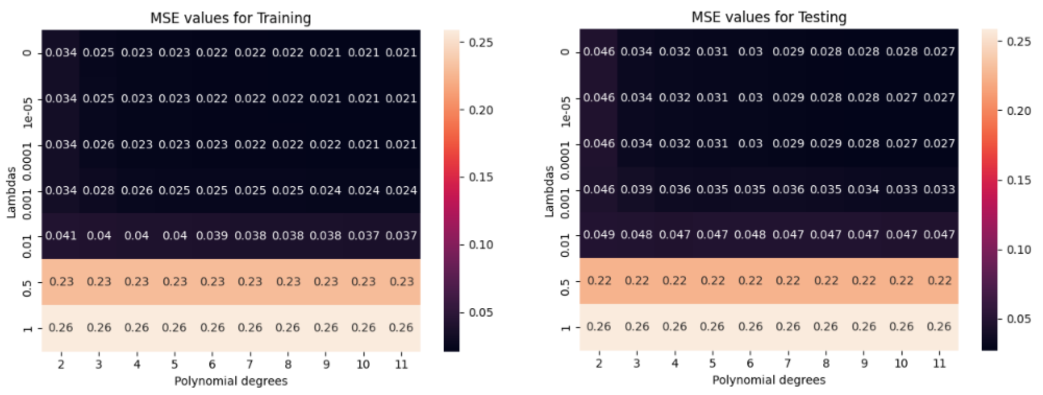


Figure 20: Heat-maps showing the behavior of training (left) and testing (right) MSE scores as lambda and polynomial degrees increase.

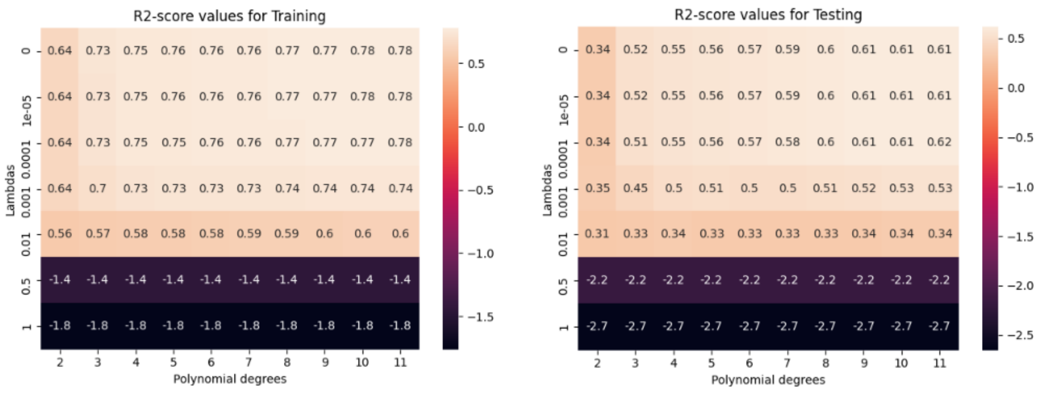


Figure 21: Heat-maps showing the behavior of training (left) and testing (right) R2-scores as lambdas and polynomial degrees increase.

Hence, as expected and similar to Ridge's model, as long as the lambda increases, the variance for higher polynomial degrees decreases compared to the OSL's model.

Lambda equals to 0,0001 is chosen for studying the training and testing MSE plot below:

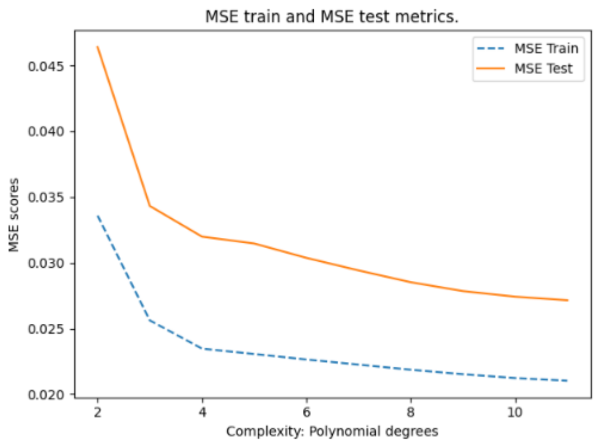


Figure 22: Line plot showing the behavior of training and testing MSE-scores for lambda=0.0001, number of samples equals 20 and polynomial degrees between 2, 12.

There is still a lower variance for a higher polynomial degree than the OLS's plot [3.2.1] and quite the same as Ridge's [3.4.2].

3.5.3 Lasso's Bias-variance trade-off with bootstrapping

It was performed a 100x bootstrapping for studying the bias-variance trade-off. For that, we set lambda=0.0001, nr_samples=20, and poly_degrees from 2 to 12:

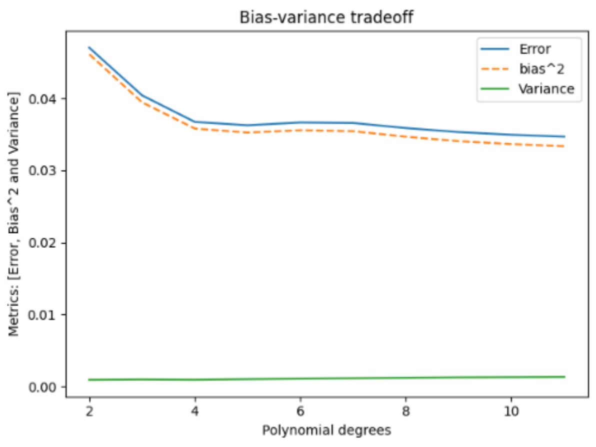


Figure 23: Line plot showing the bias-variance trade-off.

The decomposition of the bias-variance trade-off shows a lower variance for a higher polynomial degree than the OLS's [3.2.2] and Ridge's plot [3.4.3], with the same parameters.

3.5.4 Lasso's k-folds cross-validation

For nr_samples=20 and lambda=0,0001, we run the k-folds cross-validation for polynomial degrees between 2 to 12:

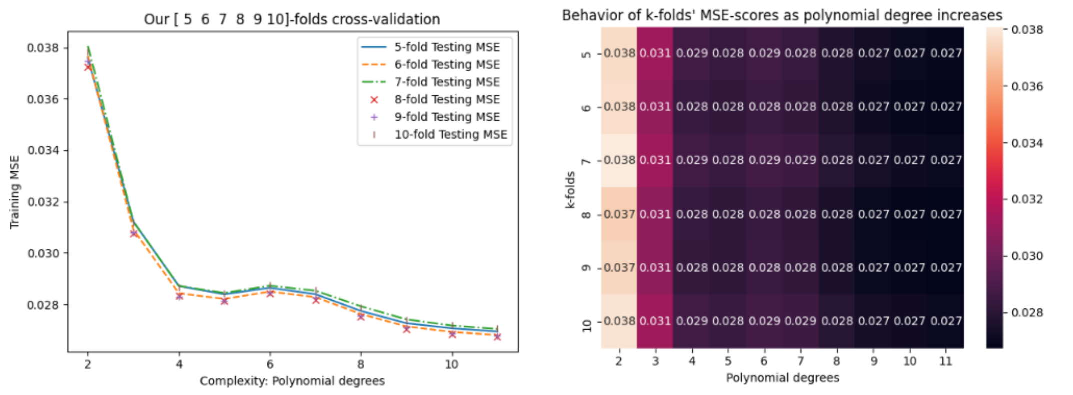


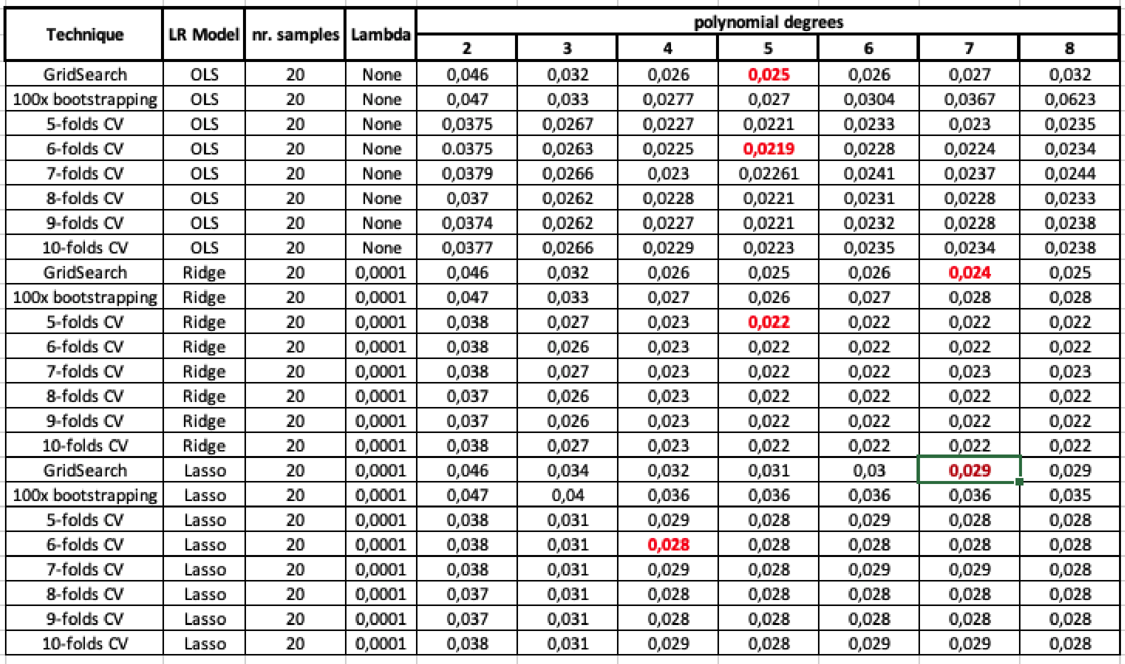
Figure 24: Left side containing line plots for different k-folds. Right side containing heat-map with the k-folds' MSE results.

Notice that the MSE values converge to its minimum for all k-folds numbers.

3.5.5 Part E - Outputs and conclusions

In this part, we studied Lasso's regression model by introducing GridSearch, bias-variance trade-offs with bootstrapping, and k-folds cross-validation. The outputs are in this report's appendix section [6.5]. The experiment can be reproduced by using the file partE.py in our GitHub repository.

The comparison among GridSearch (partA), bias-variance trade-off with bootstrapping (partB) and k-folds cross-validation (partC) is below:



Technique	LR Model	nr. samples	Lambda	polynomial degrees						
				2	3	4	5	6	7	8
GridSearch	OLS	20	None	0,046	0,032	0,026	**0,025**	0,026	0,027	0,032
100x bootstrapping	OLS	20	None	0,047	0,033	0,0277	0,027	0,0304	0,0367	0,0623
5-folds CV	OLS	20	None	0,0375	0,0267	0,0227	0,0221	0,0233	0,023	0,0235
6-folds CV	OLS	20	None	0.0375	0,0263	0,0225	**0,0219**	0,0228	0,0224	0,0234
7-folds CV	OLS	20	None	0,0379	0,0266	0,023	0,02261	0,0241	0,0237	0,0244
8-folds CV	OLS	20	None	0,037	0,0262	0,0228	0,0221	0,0231	0,0228	0,0233
9-folds CV	OLS	20	None	0,0374	0,0262	0,0227	0,0221	0,0232	0,0228	0,0238
10-folds CV	OLS	20	None	0,0377	0,0266	0,0229	0,0223	0,0235	0,0234	0,0238
GridSearch	Ridge	20	0,0001	0,046	0,032	0,026	0,025	0,026	**0,024**	0,025
100x bootstrapping	Ridge	20	0,0001	0,047	0,033	0,027	0,026	0,027	0,028	0,028
5-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	**0,022**	0,022	0,022	0,022
6-folds CV	Ridge	20	0,0001	0,038	0,026	0,023	0,022	0,022	0,022	0,022
7-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	0,022	0,022	0,023	0,023
8-folds CV	Ridge	20	0,0001	0,037	0,026	0,023	0,022	0,022	0,022	0,022
9-folds CV	Ridge	20	0,0001	0,037	0,026	0,023	0,022	0,022	0,022	0,022
10-folds CV	Ridge	20	0,0001	0,038	0,027	0,023	0,022	0,022	0,022	0,022
GridSearch	Lasso	20	0,0001	0,046	0,034	0,032	0,031	0,03	**0,029**	0,029
100x bootstrapping	Lasso	20	0,0001	0,047	0,04	0,036	0,036	0,036	0,036	0,035
5-folds CV	Lasso	20	0,0001	0,038	0,031	0,029	0,028	0,029	0,028	0,028
6-folds CV	Lasso	20	0,0001	0,038	0,031	**0,028**	0,028	0,028	0,028	0,028
7-folds CV	Lasso	20	0,0001	0,038	0,031	0,029	0,028	0,029	0,029	0,028
8-folds CV	Lasso	20	0,0001	0,037	0,031	0,028	0,028	0,028	0,028	0,028
9-folds CV	Lasso	20	0,0001	0,037	0,031	0,028	0,028	0,028	0,028	0,028
10-folds CV	Lasso	20	0,0001	0,038	0,031	0,029	0,028	0,029	0,029	0,028

Figure 25: Comparison for testing MSE scores between techniques.

The best testing MSE achieved with Lasso was 0,028 for 6-folds cross-validation, which is worse than OLS's and Ridge's performances.

Lasso regression makes it possible to increase the polynomial degrees (complexity of the model) with lower variance than OLS; then, it might be beneficial and might get better scores if we increase the model's complexity.

3.6 Part F and G of the study

This section will apply all the subjects studied before on real terrain heights. For that, we downloaded a GeoTIF file from https://earthexplorer.usgs.gov/, containing specific altitudes of a region near Stavanger in Norway, and stored it in the directory data/. The visualization of this region is plotted below:

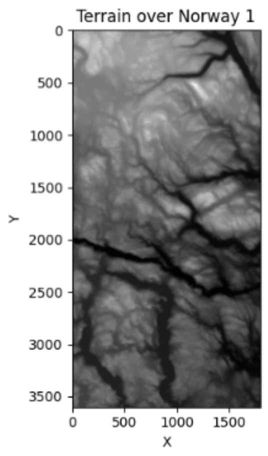


Figure 26: Terrain heights plot from a GeoTIF file.

3.6.1 OLS model on terrain data

We utilize the same polynomial transformation method used in the previous exercises to generate our explanatory variables x and y. The response variable z will be the heights extracted from the GeoTIF file downloaded. We also slice the original image to a smaller fraction to avoid expensive computational calculations and run a GridSearch to analyze the behavior of the MSE score and R2-score under the OLS model:

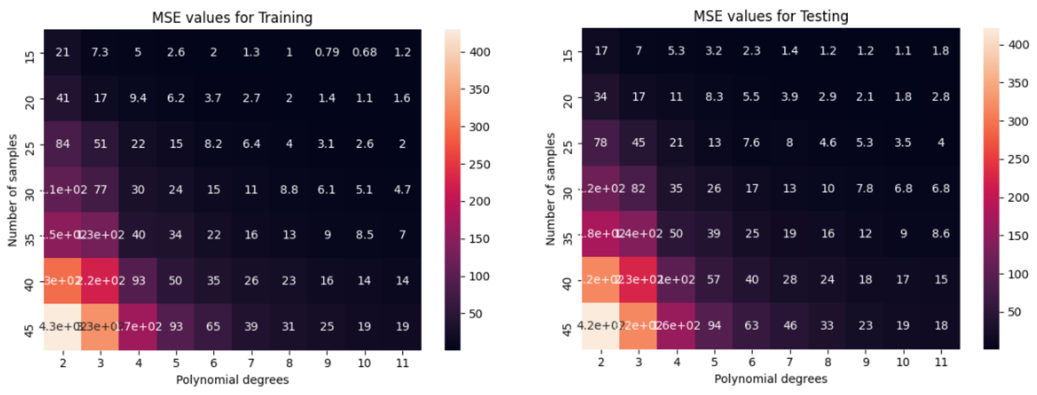


Figure 27: Heat-map with the training (left) and testing (right) MSE scores for the terrain's heights using OLS model.

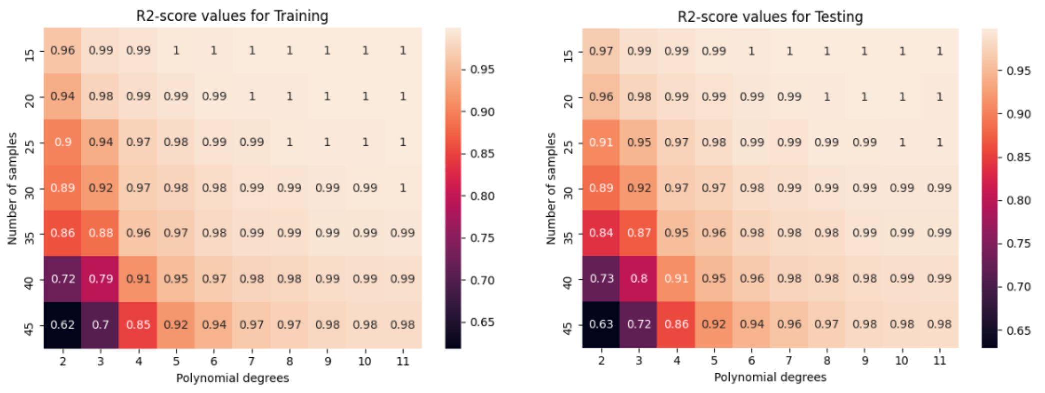


Figure 28: Heat-map with the training (left) and testing (right) R2-score for the terrain's heights using OLS model.

One can see that the predictions are more accurate when the image/region is smaller. As long the picture gets larger, the precision decreases, but with still a very high R2-score. This behavior means that our regression model does a better job predicting smaller regions than bigger ones with higher complexities. Besides, high R2-scores means that the explanatory variables do a very good job explaining the response variable, in a rate of approximately 60-90 percent in these heat-map cases.

Notice, we can also improve MSE performance for larger pictures by increasing the polynomial degree. However, it will become significantly computational expensive. Thus, for didactic propose, the size of the image is chosen at 15 nr_samples.

Following are the results for the training and testing MSE scores, the decomposition of the bias-variance trade-offs with bootstrapping and k-folds cross-validation:

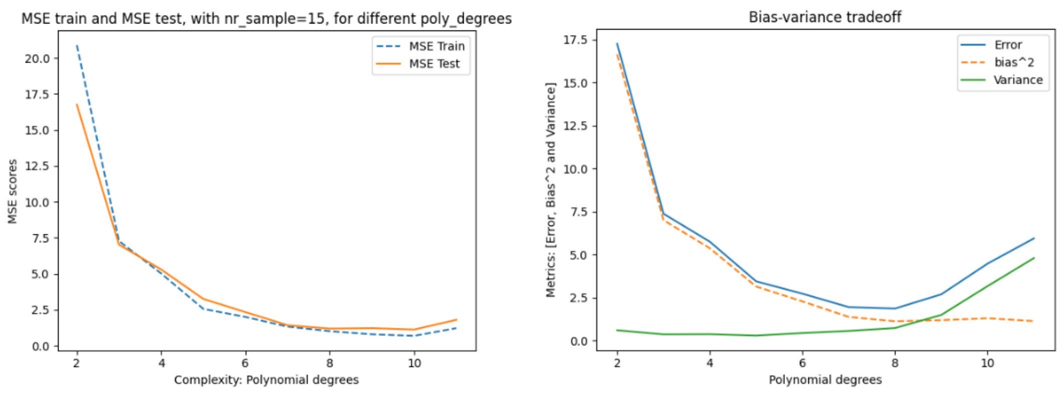


Figure 29: Left-side containing a line-plot for training and testing MSE. Right-side with the bias-variance decomposition with 100x bootstrapping.

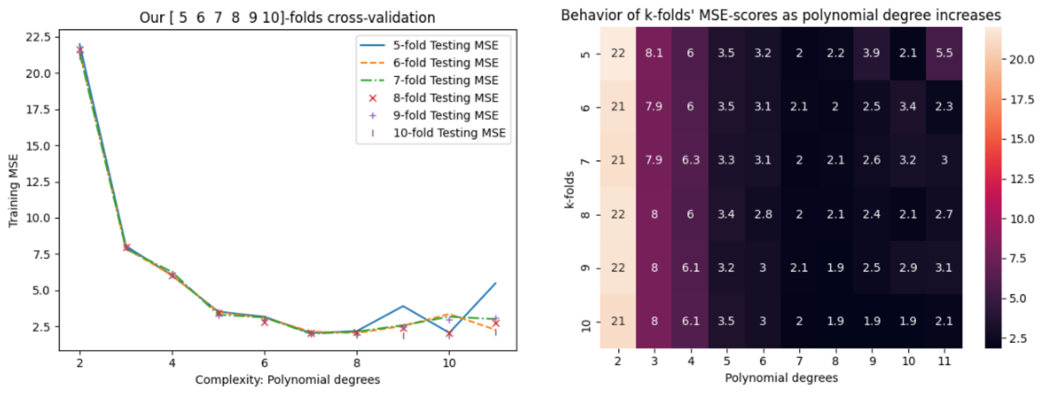
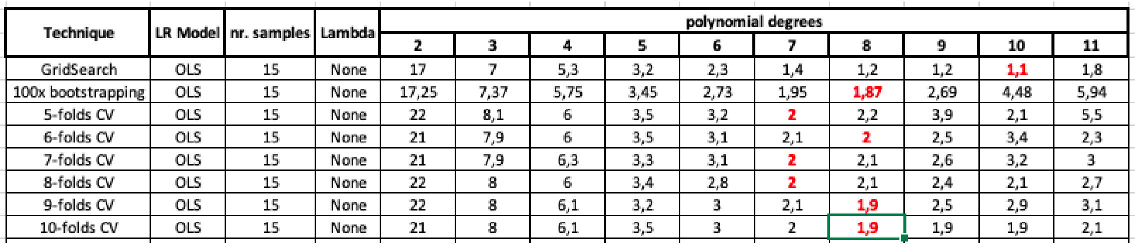


Figure 30: Left-side with k-folds' cross-validation line plots. Right-side containing k-folds' cross-validation results.

From the left-side of figure 29, the best testing MSE is 1.1 for a polynomial degree 10. However, from the right side of figure 29, this 1.1 result might be over-estimated because the bias-variance trade-off shows a significant increase in the variance of the model at polynomial degree 7. This supposed over-estimation is also confirmed by the k-folds cross-validation line-plot and heat-map, in figure 30, where the best testing MSE was obtained with a lower polynomial degree around 7 or 8.

A break-down of the study-results can be checked below:



Technique	LR Model	nr. samples	Lambda	polynomial degrees									
				2	3	4	5	6	7	8	9	10	11
GridSearch	OLS	15	None	17	7	5,3	3,2	2,3	1,4	1,2	1,2	**1,1**	1,8
100x bootstrapping	OLS	15	None	17,25	7,37	5,75	3,45	2,73	1,95	**1,87**	2,69	4,48	5,94
5-folds CV	OLS	15	None	22	8,1	6	3,5	3,2	**2**	2,2	3,9	2,1	5,5
6-folds CV	OLS	15	None	21	7,9	6	3,5	3,1	2,1	**2**	2,5	3,4	2,3
7-folds CV	OLS	15	None	21	7,9	6,3	3,3	3,1	**2**	2,1	2,6	3,2	3
8-folds CV	OLS	15	None	22	8	6	3,4	2,8	**2**	2,1	2,4	2,1	2,7
9-folds CV	OLS	15	None	22	8	6,1	3,2	3	2,1	**1,9**	2,5	2,9	3,1
10-folds CV	OLS	15	None	21	8	6,1	3,5	3	2	**1,9**	1,9	1,9	2,1

Figure 31: Break-down of the results among studies.

The conclusion one can obtain from these results is that, for a small picture/region sliced with only 15 samples, we obtain the sweet-spot of the predictions around polynomial of 7 or 8 degrees. Over that, the model starts to over-fits with a significant increase in the variance. Under that, the models under-fits because of the lack of information.

3.6.2 Ridge model on terrain data

This experiment utilizes the same polynomial transformation used before to generate the design matrix X. Also, the targets z are the heights from the GeoTIF picture. However, this research focuses on the Ridge regression, especially finding the best regularization or the optimum parameter lambda's best value.

For that, the algorithm named partFandG.py in our Github repository directory Project1/ does the calculations job for us, such that:

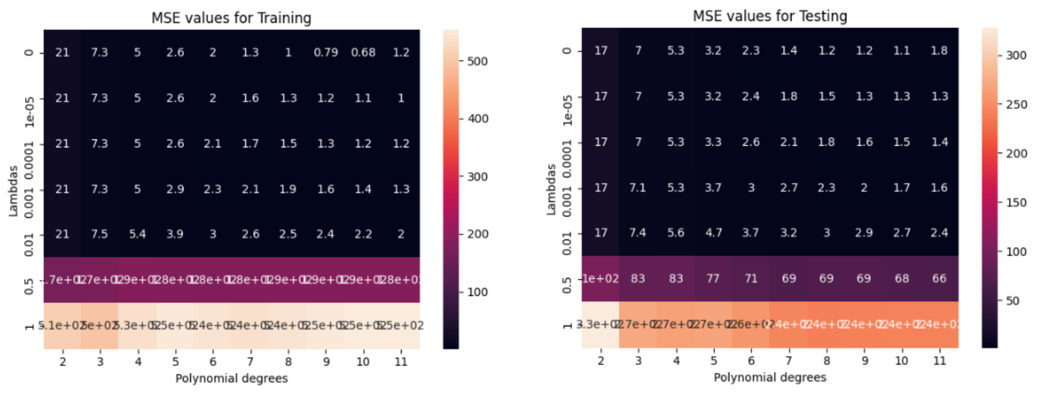


Figure 32: Left-side containing a heat-map with training MSE scores for different polynomial degrees and lambdas. Right-side containing a heat-map with testing MSE scores for different polynomial degrees and lambdas.

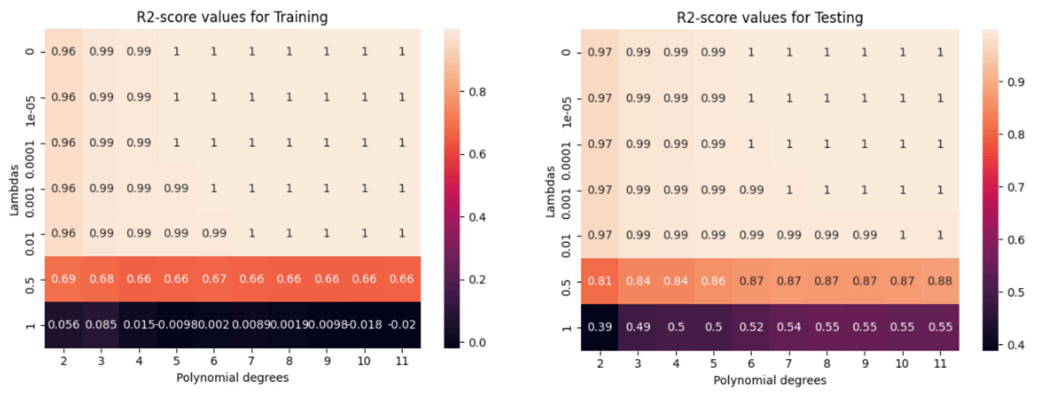


Figure 33: Left-side containing a heat-map with training R2 scores for different polynomial degrees and lambdas. Right-side containing a heat-map with testing R2 scores for different polynomial degrees and lambdas.

The significant difference here from the heat-maps for the OLS model [3.6.1] is that Ridge regression with regularization enables us to get a higher polynomial degree without over-estimate the predictions. For instance, the OSL model reaches its minimum testing scores at polynomial degrees around 8-10 for lambda 0.0001; on the other hand, the Ridge's testing score, with the same parameters, is still decreasing, showing that the variance has not increased yet.

Let us see the bias-variance decomposition for 100x bootstrapping and k-folds cross-validation:

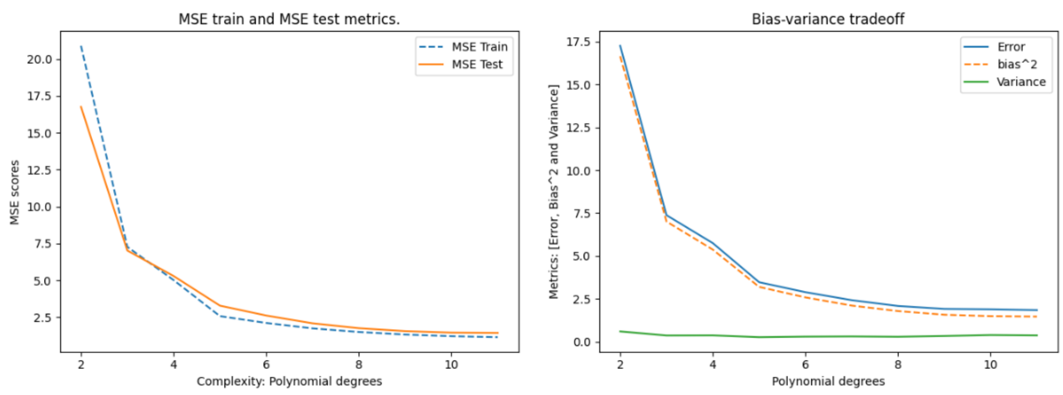


Figure 34: Left-side containing a line-plot with training and testing MSE scores for different polynomial degrees, and lambda=0.0001. Right-side containing a line-plot with the decomposition of the bias-variance with 100x bootstrapping, for different polynomial degrees, and lambda=0.0001.

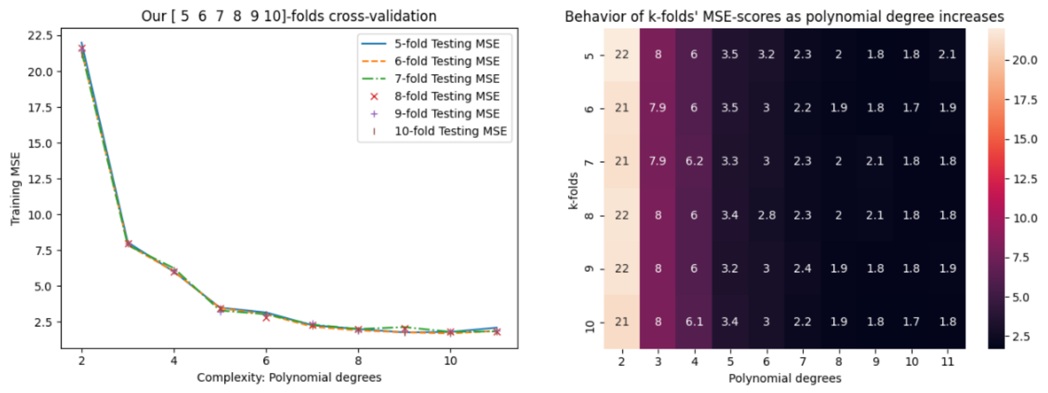
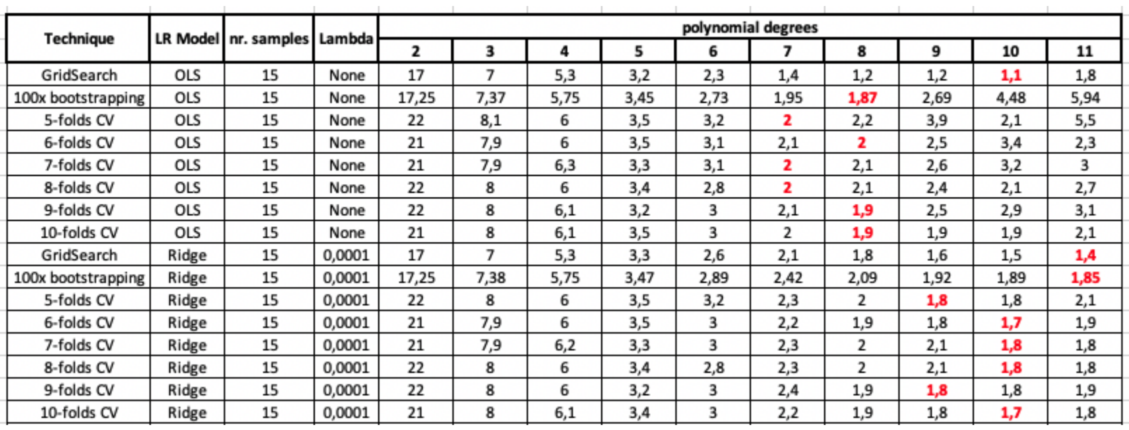


Figure 35: Left-side containing a line-plot with testing MSE scores for different polynomial degrees and k-folds, and lambdas=0.0001. Right-side containing a heat-map with testing MSE scores for different polynomial degrees and k-folds, and lambda=0.0001.

From figure 34, the variance has slowly started to increase at polynomial's degree 10, but much less than in the same circumstance at the OLS's model [3.6.1]. Besides, the Ridge's testing MSE is still descending towards its minimum at polynomial 11, in contrast to the OLS's model [3.6.1], where its minimum had already occurred.

Figure 35 confirms what was said; the scores are still going towards its minimum at polynomial degree 10-11; however, they have better metrics than the bootstrapping technique because it slightly reduces the model's complexity when dividing the samples into groups of k-folds.

Therefore, for comparison propose, a table with scores between the OLS model and Ridge model is stated as follows:



Technique	LR Model	nr. samples	Lambda	polynomial degrees									
				2	3	4	5	6	7	8	9	10	11
GridSearch	OLS	15	None	17	7	5,3	3,2	2,3	1,4	1,2	1,2	**1,1**	1,8
100x bootstrapping	OLS	15	None	17,25	7,37	5,75	3,45	2,73	1,95	**1,87**	2,69	4,48	5,94
5-folds CV	OLS	15	None	22	8,1	6	3,5	3,2	**2**	2,2	3,9	2,1	5,5
6-folds CV	OLS	15	None	21	7,9	6	3,5	3,1	2,1	**2**	2,5	3,4	2,3
7-folds CV	OLS	15	None	21	7,9	6,3	3,3	3,1	**2**	2,1	2,6	3,2	3
8-folds CV	OLS	15	None	22	8	6	3,4	2,8	**2**	2,1	2,4	2,1	2,7
9-folds CV	OLS	15	None	22	8	6,1	3,2	3	2,1	**1,9**	2,5	2,9	3,1
10-folds CV	OLS	15	None	21	8	6,1	3,5	3	2	**1,9**	1,9	1,9	2,1
GridSearch	Ridge	15	0,0001	17	7	5,3	3,3	2,6	2,1	1,8	1,6	1,5	**1,4**
100x bootstrapping	Ridge	15	0,0001	17,25	7,38	5,75	3,47	2,89	2,42	2,09	1,92	1,89	**1,85**
5-folds CV	Ridge	15	0,0001	22	8	6	3,5	3,2	2,3	2	**1,8**	1,8	2,1
6-folds CV	Ridge	15	0,0001	21	7,9	6	3,5	3	2,2	1,9	1,8	**1,7**	1,9
7-folds CV	Ridge	15	0,0001	21	7,9	6,2	3,3	3	2,3	2	2,1	**1,8**	1,8
8-folds CV	Ridge	15	0,0001	22	8	6	3,4	2,8	2,3	2	2,1	**1,8**	1,8
9-folds CV	Ridge	15	0,0001	22	8	6	3,2	3	2,4	1,9	**1,8**	1,8	1,9
10-folds CV	Ridge	15	0,0001	21	8	6,1	3,4	3	2,2	1,9	1,8	**1,7**	1,8

Figure 36: Break-down of the results among studies.

3.6.3 Lasso model on terrain data

This last practice will utilize the Lasso linear regression model with the same parameters, design matrix, and targets used in the previous sections.

For that, the algorithm named partFandG.py in our Github repository directory Project1/ does the calculations job for us, such that:

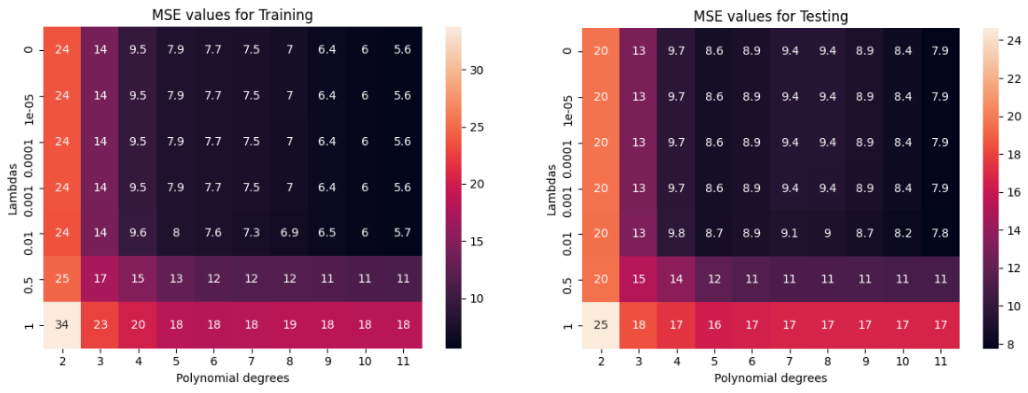


Figure 37: Left-side containing a heat-map with training MSE scores for different polynomial degrees and lambdas. Right-side containing a heat-map with testing MSE scores for different polynomial degrees and lambdas.

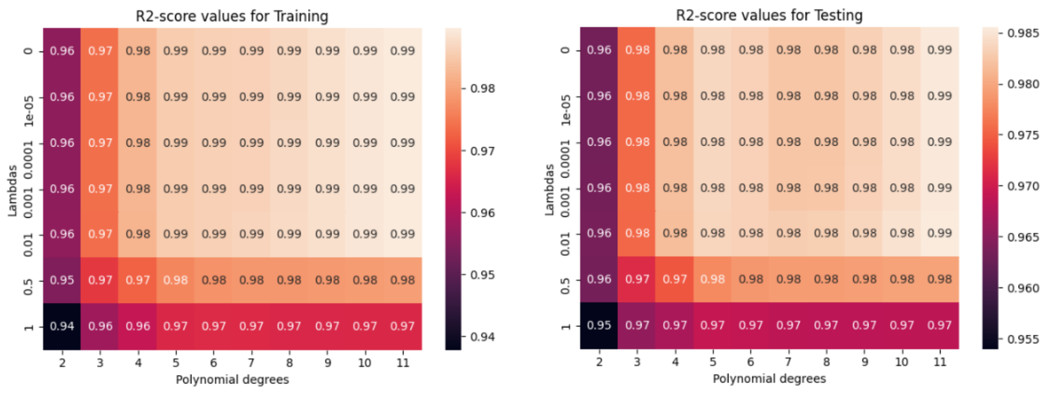


Figure 38: Left-side containing a heat-map with training R2 scores for different polynomial degrees and lambdas. Right-side containing a heat-map with testing R2 scores for different polynomial degrees and lambdas.

Lasso did not perform as good as OLS or Ridge did for polynomials between 2 and 11 because the testing MSE metrics are much higher. However, the R2-scores are still higher, inferring that the explanatory variables explain the response variables reasonably.

Let us see the bias-variance decomposition for 100x bootstrapping and k-folds cross-validation:

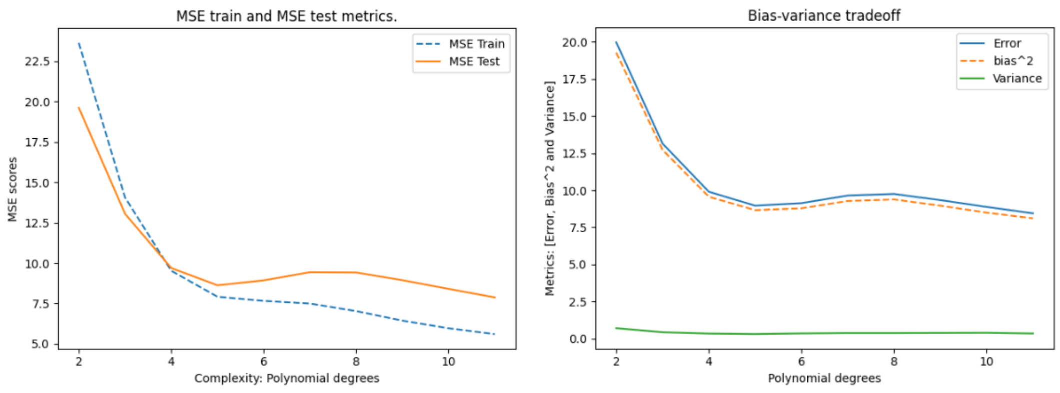


Figure 39: Left-side containing a line-plot with training and testing MSE scores for different polynomial degrees, and lambda=0.0001. Right-side containing a line-plot with the decomposition of the bias-variance with 100x bootstrapping, for different polynomial degrees, and lambda=0.0001.

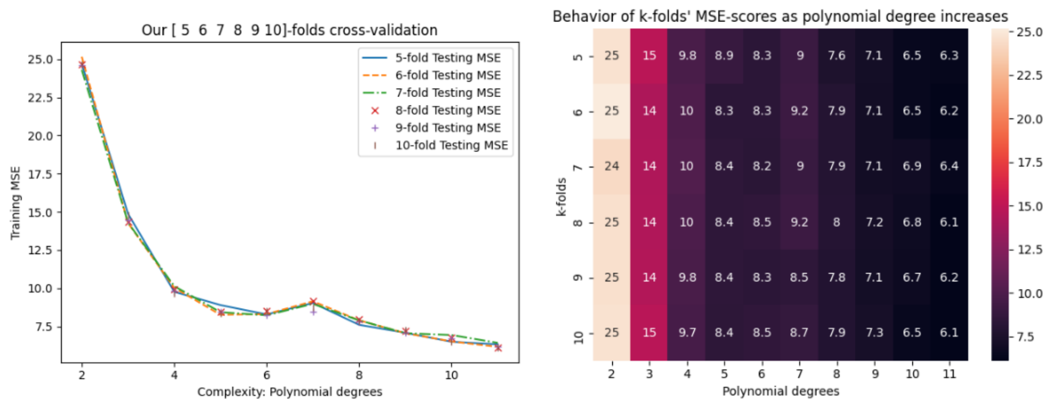
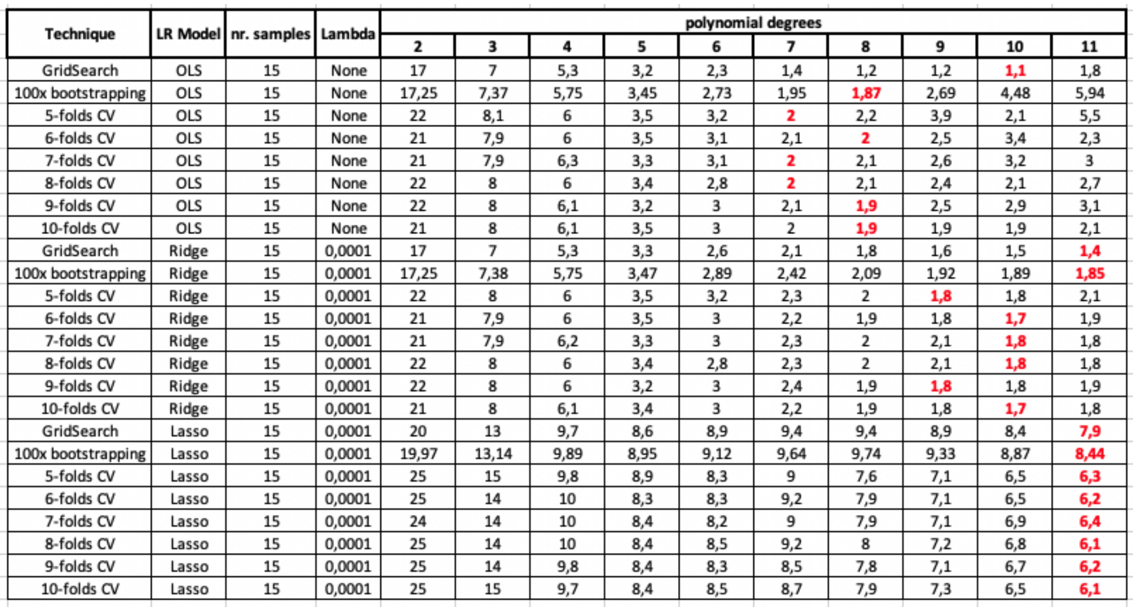


Figure 40: Left-side containing a line-plot with testing MSE scores for different polynomial degrees and k-folds, and lambdas=0.0001. Right-side containing a heat-map with testing MSE scores for different polynomial degrees and k-folds, and lambda=0.0001.

Both techniques show that the Lasso has the potential to get a better testing MSE if the polynomial degree increases. Figure 39, right, shows that the model's variance has not increased significantly for any of the polynomial's degree presented. Thus, the model can receive higher polynomials than Ridge's or OLS's models before starts to over-fit.

Below are the final break-down of the testing MSE for the three linear regression techniques:



Technique	LR Model	nr. samples	Lambda	polynomial degrees									
				2	3	4	5	6	7	8	9	10	11
GridSearch	OLS	15	None	17	7	5,3	3,2	2,3	1,4	1,2	1,2	**1,1**	1,8
100x bootstrapping	OLS	15	None	17,25	7,37	5,75	3,45	2,73	1,95	**1,87**	2,69	4,48	5,94
5-folds CV	OLS	15	None	22	8,1	6	3,5	3,2	**2**	2,2	3,9	2,1	5,5
6-folds CV	OLS	15	None	21	7,9	6	3,5	3,1	2,1	**2**	2,5	3,4	2,3
7-folds CV	OLS	15	None	21	7,9	6,3	3,3	3,1	**2**	2,1	2,6	3,2	3
8-folds CV	OLS	15	None	22	8	6	3,4	2,8	**2**	2,1	2,4	2,1	2,7
9-folds CV	OLS	15	None	22	8	6,1	3,2	3	2,1	**1,9**	2,5	2,9	3,1
10-folds CV	OLS	15	None	21	8	6,1	3,5	3	2	**1,9**	1,9	1,9	2,1
GridSearch	Ridge	15	0,0001	17	7	5,3	3,3	2,6	2,1	1,8	1,6	1,5	**1,4**
100x bootstrapping	Ridge	15	0,0001	17,25	7,38	5,75	3,47	2,89	2,42	2,09	1,92	1,89	**1,85**
5-folds CV	Ridge	15	0,0001	22	8	6	3,5	3,2	2,3	2	**1,8**	1,8	2,1
6-folds CV	Ridge	15	0,0001	21	7,9	6	3,5	3	2,2	1,9	1,8	**1,7**	1,9
7-folds CV	Ridge	15	0,0001	21	7,9	6,2	3,3	3	2,3	2	2,1	**1,8**	1,8
8-folds CV	Ridge	15	0,0001	22	8	6	3,4	2,8	2,3	2	2,1	**1,8**	1,8
9-folds CV	Ridge	15	0,0001	22	8	6	3,2	3	2,4	1,9	**1,8**	1,8	1,9
10-folds CV	Ridge	15	0,0001	21	8	6,1	3,4	3	2,2	1,9	1,8	**1,7**	1,8
GridSearch	Lasso	15	0,0001	20	13	9,7	8,6	8,9	9,4	9,4	8,9	8,4	**7,9**
100x bootstrapping	Lasso	15	0,0001	19,97	13,14	9,89	8,95	9,12	9,64	9,74	9,33	8,87	**8,44**
5-folds CV	Lasso	15	0,0001	25	15	9,8	8,9	8,3	9	7,6	7,1	6,5	**6,3**
6-folds CV	Lasso	15	0,0001	25	14	10	8,3	8,3	9,2	7,9	7,1	6,5	**6,2**
7-folds CV	Lasso	15	0,0001	24	14	10	8,4	8,2	9	7,9	7,1	6,9	**6,4**
8-folds CV	Lasso	15	0,0001	25	14	10	8,4	8,5	9,2	8	7,2	6,8	**6,1**
9-folds CV	Lasso	15	0,0001	25	14	9,8	8,4	8,3	8,5	7,8	7,1	6,7	**6,2**
10-folds CV	Lasso	15	0,0001	25	15	9,7	8,4	8,5	8,7	7,9	7,3	6,5	**6,1**

Figure 41: Break-down of the results among studies.

4 Conclusion

After an exhaustive study of the various regression models and machine learning techniques, we are prepared to decide which method is most appropriate for each type of circumstance.

Starting with Franke's data-set, our regression models could predict the surface's height with outstanding precision. In our first test [3.1] with the OLS model, we could obtain coefficients of determination at a rate of between 65 and 77 percent for both training and testing predictions. The MSE's score was also pretty good as lower as 0.019 for training and 0.035 for testing sets, but indicating under-fitting in which adjusts would be necessary. For that, part B [3.2] presented a GridSearch where we could visualize the testing MSE score's behavior when the complexities increases [3.2.1]. For didactic propose, a bench-mark parameter of 20 samples was defined and then applied different polynomial transformation degrees from 2 to 9. We could visualize an improvement of the testing MSE from 0.035 to 0.025 at the polynomial degree 5 [3.2.1] with that bench-marked parameter. The line-plot in [3.2.1] showed a relevant detachment between training and testing MSE rates when fitting the model with polynomial degree 6 or higher, meaning a plausible over-fitting. To check that in more detail, a 100x bootstrapping re-sampling technique combined with bias-variance trade-off decomposition was plotted [3.2.2] and confirmed a rapid growth of the variance of the model after polynomial degree 6. Later, the 5-folds cross-validation [3.3.1] successfully attenuated that spike of the variance of the model for polynomials above 5-6 degrees, and it did decrease the MSE score from 0.025 to 0.021, our best score here at the polynomial degree 5.

Now, Ridge regression and its regularization parameter lambda took place in the study of Franke's function. The aim was to beat the previous testing MSE (0.021) by implementing different lambda values. After the GridSearch between several lambdas and polynomial degrees, we picked lambda=0.0001 as our bench-mark parameter, which obtained a 0.024 testing MSE metric at the polynomial degree 7 [3.4.2]. The line-plot of training and testing MSE [3.4.2] showed that, differently from the OLS model, the model's over-fitting appeared to come later. The bias-variance trade-off with 100x bootstrapping [3.4.3] confirmed that suspicion and registered that the variance did come later and with less power than in the OLS model [3.2.2]. This effect proved that the regularization of the Ridge regression had influenced the results. We could provide more complexity (higher polynomial degree) to the model without suffering much over-fitting, but the best testing MSE obtained was higher/worse than what was obtained for the OLS model. Finally, the 5-fold cross-validation procedure [3.4.4] could get 0.022, at polynomial degree 5, but not better than 0.0219 obtained before.

For Lasso regression under Franke's function, we have done the same procedure as Ridge. First, GridSearch to find the best lambda regularization value, which ended up been 0.0001 [3.5.2]. Line-plot with training and testing MSE [3.5.2] revealed that the metrics were going to its minimum even for higher polynomial degrees, inferring that the over-fitting had not come. The bias-variance trade-off with 100x bootstrapping [3.5.3] validated this assumption; there was no sign of an increasing variance until polynomial 11. The same was observed for 6-folds cross-validation [3.5.4], where the best score was 0.028 for polynomial degree 4, which was not better than 0.0219 obtained before.

In its turn, we started to analyze the GeoTIF image containing heights of a region near Stavanger. The aim here was to fit the regression models with the design matrix of x and y between 0 and 1, and after predicting the heights. We noticed that the picture needed to be sliced to a smaller piece for didactic propose and avoid expensive computational calculation, then the bench-mark of 20 samples was selected after a GridSearch [3.6.1] when it was obtained the testing MSE of 1.1 for polynomial degree 10 under the OLS model. Later, the training and testing MSE line-plot [3.6.1] combined with the bias-variance with 100x bootstrapping revealed that a likely over-fitting could occur around polynomial degree 8; therefore, the score obtained previously could have been over-estimated. Next, the k-fold cross-validation [3.6.1] was realized but got much higher testing MSE, around 2, than before.

Moreover, the Ridge regression with the same parameters as before got a score of 1.4 [3.6.2] at polynomial degree 11, for lambda equals 0.0001, our bench-marked parameter. We noticed, by training and testing line-plot [3.6.2] and bias-variance with bootstrapping [3.6.2], that Ridge's regularization avoided the variance to appear in contrast to the OLS model when was possible to use higher complexity in terms of polynomial degree. Even though the Ridge method's testing MSE score (1.4) was higher than the OLS (1.1), that higher score might be more reliable because OLS's lower score was confirmed as over-estimated compared to the Ridge's higher score that was not affected by the variance. Lastly, the 6-fold cross-validation [3.6.2] did not a better job and got scores around 1.7 for polynomial degree 10.

In the end, Lasso regression with lambda equals 0.0001 took place and got much higher testing MSE scores for all experiments, for instance, GridSearch (7.9 for polynomial degree 11) [3.6.3], bias-variance with bootstrapping (8.44) [3.6.3], k-fold cross-validation (6.1) [3.6.3]. Nevertheless, these higher scores could occur because of the Lasso's regularization, which, as happened with Ridge, avoided the variance to rise.

After all, one can conclude that the bias-variance trade-off is crucial when performing predictions because they determine if the prediction's metrics are under- or over-estimated or influenced by under- or over-fitting. Therefore, after knowing this trade-off, we can pick the best pre-processing, model, regularization, predictions, and metrics. In our cases, for Franke's function, the best approach was 5-folds cross-validation with polynomial degree 5 for 15 number of samples under the OLS model. For the GeoTIF image, the best procedure was polynomial degree 11 for 20 samples under the Ridge method with 0.0001 regularization. Also, for the GeoTIF image, it could have had a better score if a higher polynomial degree was applied under Ridge or Lasso methods. Additionally, if one wanted to predict a higher area (higher number of samples) for the GeoTIF image, it would have been wise to use a higher polynomial degree combined with Ridge or Lasso models with regularization because it slowed the variance to develop, then, the same for over-fitting. These tests could be an excellent topic for future experiments and improvements.

5 Bibliography

[1] T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning data mining, inference, and prediction.* **2nd edition**. Springer; New York, USA. 2017.

[2] Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning.* **2nd edition**. Packt; Birmingham, UK. 2017.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly Media; Sebastopol, CA. 2017.

6 Appendix

6.1 Output - Part A

```
Betas for polynomial of [2] degree:
[ 1.11208279 -0.24465572 -0.18711398 -0.02279257
  0.18029852 -0.10002226]

CI for betas:  [0.22619356 0.22131374 0.17460161
                0.19242439 0.11710238 0.15691339]

Training MSE:  0.02765091091053581
Training R2-score:  0.6502230601066621

Test MSE:  0.06916999928486468
Test R2-score:  0.5586077637643143

Betas for polynomial of [3] degree:
[ 1.01985474 -0.08135026  0.15296646 -0.41991135  0.29825074
 -1.14321385 0.19123732  0.10076225 -0.19768809  0.75785779]

CI for betas:  [0.37963427 0.68635664 0.48508963 1.10371127
                0.78989626 0.92864678 0.55350841 0.51302814
                0.36837101 0.54929345]

Training MSE:  0.02299523348619776
Training R2-score:  0.7091161869148273

Test MSE:  0.052779893268975725
Test R2-score:  0.6631974069808669

Betas for polynomial of [4] degree:
[ 0.72676751  1.16324899  0.41344867 -5.09992689 -0.79495083
 -1.52064403  6.18192618  1.57015774  0.50569783  0.921933
 -2.55379606 -0.58105319 -0.39244227 -0.15886949  0.00760222]

CI for betas:  [0.63807188 1.65655155 1.16351475 4.54921129
                3.30655009 3.65438871 5.2233967  3.71869945
                3.13453628 4.7979061  2.21813435 1.45666743
                1.57547549 1.19535724 2.198263]

Training MSE:  0.02048304759120491
Training R2-score:  0.7408946949588061

Test MSE:  0.04332728430232288
Test R2-score:  0.7235170289729023

Betas for polynomial of [5] degree:
[ 0.52503804  1.7896095   0.74886353 -6.7249159  -3.65470155
 -1.05890768 7.19828896  6.27610772  5.80255991 -3.78738869
 -1.8110568  -4.64639806 -3.30915019 -5.29258591  7.40861838
 -0.73377809  1.31055844  0.95997903 0.72490794  1.95374315
 -3.46255493]
```

```
CI for betas:  [ 1.02667143  3.82242848  2.4961594  17.64198866
                  10.55603815 11.31509895 36.53792412 18.92199964
                  16.54165772 24.40583381 35.45255739 15.52938946
                  15.39734153 12.89825879 24.55238258 12.9965755
                  5.33125202  4.73094118 5.33351166  4.55626702
                  9.25419634]

Training MSE:  0.01978457394650227
Training R2-score:  0.7497302076415779

Test MSE:  0.035345653129498146
Test R2-score:  0.7744499488602161
```

6.2 Output - Part B

```
Best MSE for training:  0.012618652852079882
 in Poly_degree=9 and Parameter2=10
Best MSE for testing:  0.020786655387886453
 in Poly_degree=7 and Parameter2=45
Best R2-score for training:  0.842395182662469
 in Poly_degree=9 and Parameter2=10
Best R2-score for testing:  0.7830350450510821
 in Poly_degree=8 and Parameter2=30

Complexity:  2
Error:  0.04777531840828764
Bias^2:  0.046727755508959856
Var:  0.0010475628993277819

0.04777531840828764 >= 0.046727755508959856 +
0.0010475628993277819 = 0.04777531840828764

Complexity:  3
Error:  0.033012563703805065
Bias^2:  0.031889160397044404
Var:  0.001123403306760654

0.033012563703805065 >= 0.031889160397044404 +
0.001123403306760654 = 0.03301256370380506

Complexity:  4
Error:  0.027748056591498094
Bias^2:  0.02626127731579172
Var:  0.0014867792757063722

0.027748056591498094 >= 0.02626127731579172 +
0.0014867792757063722 = 0.02774805659149809

Complexity:  5
Error:  0.027092529195362
Bias^2:  0.02497998774788366
Var:  0.002112541447478341

0.027092529195362 >= 0.02497998774788366 +
0.002112541447478341 = 0.027092529195362

Complexity:  6
Error:  0.030472252108762622
Bias^2:  0.02660347771746179
Var:  0.0038687743913008305

0.030472252108762622 >= 0.02660347771746179 +
0.0038687743913008305 = 0.03047225210876262

Complexity:  7
Error:  0.036787942724237695
Bias^2:  0.02710689646030625
Var:  0.00968104626393144

0.036787942724237695 >= 0.02710689646030625 +
0.00968104626393144 = 0.03678794272423769

Complexity:  8
Error:  0.06235950744943284
Bias^2:  0.03517340873671558
Var:  0.027186098712717266

0.06235950744943284 >= 0.03517340873671558 +
0.027186098712717266 = 0.06235950744943285
```

6.3 Output - Part C

```
Testing MSE for our k-folds CV:
[[0.03752289503694306, 0.026736084181286545, 0.022739810292144353,
0.022119731889857462, 0.023356740172061345, 0.023082680410993376,
0.023586417653293324], [0.0375509737755547, 0.026322819628640415,
0.022593128768955733, 0.02192151740901438, 0.02285109778696126,
0.022430121342698898, 0.023414805824800215], [0.03793399037866348,
0.02669092218981462, 0.0230030713435502, 0.022615520542590594,
0.024185331739111516, 0.023730620855473283, 0.024498579659280328],
[0.03707488769110362, 0.026252015296301955, 0.022823846456879994,
0.022181951508474837, 0.023196234120289143, 0.022877457111239244,
0.02334901206176O925], [0.037421275709054985,
0.026289845982373147, 0.02273598996184378, 0.022120697941251925,
0.023214895399550028, 0.022830115455658734, 0.02384505629015149],
[0.0377108482731467, 0.02661361158251626, 0.02294086500110492,
0.02237068430990173, 0.02355903492206367, 0.02343250228713719,
0.023806144016107766]]

 Testing MSE for SkLearn k-folds CV:
 [[0.037275376051167125, 0.025811676895413194,
 0.022829918290127666, 0.022494184544718295, 0.023714945646574555,
 0.02360490842584482, 0.024937337553679127], [0.03733098735918176,
 0.025868668742065476, 0.02268893651976892, 0.02216642664073321,
 0.023533933294930743, 0.023566796887507336,
 0.024754820782331752], [0.03697329271098665,
 0.025825670906074998, 0.02280279269246253, 0.022327081487471027,
 0.023494624439201608, 0.023568934655013897, 0.02464452638522858],
 [0.0368804588347594, 0.02585088507665252, 0.022773017385509957,
 0.02220069943977543, 0.023424809234379503, 0.02338563847719166,
 0.02427984886192284], [0.037087305304568954, 0.02594794363986065,
 0.02285645268067041, 0.022559127433098382, 0.02333657667646123,
 0.023765890799842673, 0.023824616485779635],
 [0.03714943429877249, 0.025939747348600422, 0.02284034629575985,
 0.022288199785927894, 0.023038736638887128, 0.0234324904068742,
 0.023499112968981897]]
```

6.4 Output - Part D

```
Best MSE for training:  0.014973898122966806
 in Poly_degree=11 and Parameter2=0
Best MSE for testing:  0.02435696666190417
 in Poly_degree=9 and Parameter2=0.001
Best R2-score for training:  0.8400887900285098
 in Poly_degree=11 and Parameter2=0
Best R2-score for testing:  0.6556696013164707
 in Poly_degree=9 and Parameter2=0.001

Complexity:  2
Error:  0.04777368838726391
Bias^2:  0.04672624614544145
Var:  0.0010474422418224657

0.04777368838726391 >= 0.04672624614544145 +
0.0010474422418224657 = 0.04777368838726392

Complexity:  3
Error:  0.03300755137989374
Bias^2:  0.031886423152534545
Var:  0.0011211282273592006

0.03300755137989374 >= 0.031886423152534545 +
0.0011211282273592006 = 0.033007551379893744

Complexity:  4
Error:  0.02783587498420162
Bias^2:  0.026397980587455162
Var:  0.001437894396746465

0.02783587498420162 >= 0.026397980587455162 +
0.001437894396746465 = 0.027835874984201626

Complexity:  5
Error:  0.026784756880995676
Bias^2:  0.025124063566186093
Var:  0.0016606933148095755

0.026784756880995676 >= 0.025124063566186093 +
0.0016606933148095755 = 0.02678475688099567

Complexity:  6
```

```
Error:  0.027570705826600178
Bias^2:  0.02570850267377949
Var:  0.0018622031528206836

0.027570705826600178 >= 0.02570850267377949 +
0.0018622031528206836 = 0.027570705826600175

Complexity:  7
Error:  0.028851819728435635
Bias^2:  0.02632441798171157
Var:  0.0025274017467240622

0.028851819728435635 >= 0.02632441798171157 +
0.0025274017467240622 = 0.028851819728435632

Complexity:  8
Error:  0.02867190848891334
Bias^2:  0.02562953696538649
Var:  0.0030423715235268468

0.02867190848891334 >= 0.02562953696538649 +
0.0030423715235268468 = 0.028671908488913336

Complexity:  9
Error:  0.02812614019578702
Bias^2:  0.024619289839237838
Var:  0.0035068503565491836

0.02812614019578702 >= 0.024619289839237838 +
0.0035068503565491836 = 0.028126140195787023

Complexity:  10
Error:  0.028933452745873535
Bias^2:  0.02456528604636401
Var:  0.004368166699509529

0.028933452745873535 >= 0.02456528604636401 +
0.004368166699509529 = 0.02893345274587354

Complexity:  11
Error:  0.030928703474881036
Bias^2:  0.025197124414327265
Var:  0.005731579060553769

0.030928703474881036 >= 0.025197124414327265 +
0.005731579060553769 = 0.030928703474881032

Testing MSE for our k-folds CV:
[[0.037522827050825644, 0.02673549752682991, 0.02273351471361173,
0.02178066990822509, 0.022077766461017355, 0.022382865127000516,
0.02233426636097547, 0.022225196880282256, 0.022316015275655764,
0.022577860806531506],
[0.037550932517306145, 0.026323166575039025, 0.022604100051908726,
0.021695830402208988, 0.02188328619026267, 0.022050467957100476,
0.021971118032649586, 0.021872138806583008, 0.0218984023263933,
0.022043451492261296],
[0.0379338711005646, 0.026690804192836693, 0.02299501716836124,
0.022096046867626156, 0.022356563928446562, 0.02257546335322385,
0.022523759712616383, 0.02248562510324279, 0.022592247670643946,
0.022800274653768167],
[0.03707488350203132, 0.026251544158332595, 0.022794791320869673,
0.02187450445895693, 0.022125396988214238, 0.022341706990107096,
0.02224443904562344, 0.02213694895633173, 0.022212926860752907,
0.02241917322801604],
[0.03742121983593963, 0.026289911882486874, 0.022728037699291187,
0.021838172075132414, 0.021999242427901824, 0.02217938270766194,
0.022146727016549336, 0.022137213751147668, 0.02227825190430999,
0.022516190847307112],
[0.037710759601364136, 0.026613909273059223, 0.02293448983166308,
0.02196640505011279, 0.02216711142008485, 0.022381410262259017,
0.022376037958903822, 0.022403578586954676, 0.022577101808419216,
0.02282420851175006]]
```

6.5 Output - Part E

```
Best MSE for training:  0.020815089321993954
 in Poly_degree=11 and Parameter2=0
Best MSE for testing:  0.027153740514381207
 in Poly_degree=11 and Parameter2=0.0001
Best R2-score for training:  0.7777087775133598
 in Poly_degree=11 and Parameter2=0
Best R2-score for testing:  0.6161320731415316
 in Poly_degree=11 and Parameter2=0.0001

Complexity:  2
Error:  0.047056556917836195
Bias^2:  0.04614077748776133
Var:  0.0009157794300748681
0.047056556917836195 >= 0.04614077748776133 +
0.0009157794300748681 = 0.0470565569178362

Complexity:  3
Error:  0.040437990120671494
Bias^2:  0.039470261001576704
Var:  0.0009677291190947955
0.040437990120671494 >= 0.039470261001576704 +
0.0009677291190947955 = 0.0404379901206715

Complexity:  4
Error:  0.03674700594167662
Bias^2:  0.03582273728703385
Var:  0.0009242686546427747
0.03674700594167662 >= 0.03582273728703385 +
0.0009242686546427747 = 0.036747005941676625

Complexity:  5
Error:  0.03628236487598001
```

```
Bias^2:  0.03527072068803944
Var:  0.001011644187940567
0.03628236487598001 >= 0.03527072068803944 +
0.001011644187940567 = 0.03628236487598001

Complexity:  6
Error:  0.036670928752274935
Bias^2:  0.035589517124217755
Var:  0.0010814116280571794
0.036670928752274935 >= 0.035589517124217755 +
0.0010814116280571794 = 0.036670928752274935

Complexity:  7
Error:  0.03660229186453423
Bias^2:  0.03546180292913997
Var:  0.001140488935394264
0.03660229186453423 >= 0.03546180292913997 +
0.001140488935394264 = 0.03660229186453423

Complexity:  8
Error:  0.03589488114556467
Bias^2:  0.03469265133174894
Var:  0.001202229813815734
0.03589488114556467 >= 0.03469265133174894 +
0.001202229813815734 = 0.035894881145564675

Complexity:  9
Error:  0.0353361777693787
Bias^2:  0.034072099534263475
Var:  0.001264078235115225
0.0353361777693787 >= 0.034072099534263475 +
0.001264078235115225 = 0.0353361777693787

Complexity:  10
Error:  0.03496513423113344
Bias^2:  0.03366935293245251
Var:  0.0012957812986809257
0.03496513423113344 >= 0.03366935293245251 +
0.0012957812986809257 = 0.034965134231133434

Complexity:  11
Error:  0.03470128789005454
Bias^2:  0.03337406035902104
Var:  0.0013272275310335035
0.03470128789005454 >= 0.03337406035902104 +
0.0013272275310335035 = 0.03470128789005454

Testing MSE for our k-folds CV:
[[0.03759351703862426, 0.03122261196611628,
0.028713920116966656, 0.028393225244900695,
0.028644663505300004, 0.028398179846876558,
0.027744726710777462, 0.027269040709460735,
0.027068144672985787, 0.026943893918675632],
[0.03766379550554178, 0.0309122419193343,
0.02843198436546307, 0.028211162451805234,
0.028497435681941446, 0.028281408782233647,
0.027624486739702872, 0.027143215910842528,
0.026928052261408052, 0.026805373121615354],
[0.03805522472068413, 0.03122364479809464,
0.02871538265324863, 0.028442243995358916,
0.02872595501963988, 0.028532141059110536,
0.027925151480206044, 0.027416622015769814,
0.027176655160317146, 0.027041878216863206],
[0.03722889556958513, 0.030764550584648363,
0.028342234957966228, 0.028146941856436077,
0.02842174204489173, 0.028183468399891866,
0.02752703868315967, 0.027042949364230746,
0.0268427721822105, 0.026736187606278775],
[0.03747566074039861, 0.030779803383477244,
0.02829457447034452, 0.028142697782813045,
0.02846900054321662, 0.028251001888283167,
0.027558544483808334, 0.02708731028093996,
0.02686991874575781, 0.02677732053681609],
[0.03781132715901417, 0.0311861283863454,
0.02866207635232616, 0.028438735736313614,
0.028736624067051554, 0.028548639617067505,
0.02789326892140421, 0.02739218042016976,
0.02718224204447685, 0.02706913680564224]]
```

6.6 Output - Part F and G

```
Best MSE for training:  0.6798353911154782
 in Poly_degree=10 and Parameter2=15
Best MSE for testing:  1.116780111855389
 in Poly_degree=10 and Parameter2=15
Best R2-score for training:  0.9987455410749885
 in Poly_degree=10 and Parameter2=15
Best R2-score for testing:  0.9979126742131896
 in Poly_degree=10 and Parameter2=15

Complexity:  2
Error:  17.252613822634505
Bias^2:  16.643206894718503
Var:  0.6094069279160882

17.252613822634505 >= 16.643206894718503 +
0.6094069279160882 = 17.25261382263459

Complexity:  3
Error:  7.3779463697303775
Bias^2:  7.006335560992514
Var:  0.37161080873786995

7.3779463697303775 >= 7.006335560992514 +
0.37161080873786995 = 7.377946369730384

Complexity:  4
```

```
Error:  5.756077458310136
Bias^2:  5.375404000016785
Var:  0.3806734582933054

5.756077458310136 >= 5.375404000016785 +
0.3806734582933054 = 5.756077458310091

Complexity:  5
Error:  3.452291716815451
Bias^2:  3.152596098751146
Var:  0.2996956180643068

3.452291716815451 >= 3.152596098751146 +
0.2996956180643068 = 3.4522917168154525

Complexity:  6
Error:  2.7359103941784317
Bias^2:  2.28669150482573
Var:  0.44921888935271803

2.7359103941784317 >= 2.28669150482573 +
0.44921888935271803 = 2.7359103941784477

Complexity:  7
Error:  1.9547215946113883
Bias^2:  1.3843455588431441
Var:  0.5703760357682643

1.9547215946113883 >= 1.3843455588431441 +
0.5703760357682643 = 1.9547215946114085

Complexity:  8
Error:  1.87107045032927
Bias^2:  1.132576333730995
Var:  0.7384941165982778

1.87107045032927 >= 1.132576333730995 +
0.7384941165982778 = 1.871070450329273

Complexity:  9
Error:  2.6987449903665146
Bias^2:  1.1976700918112484
Var:  1.5010748985552993

2.6987449903665146 >= 1.1976700918112484 +
1.5010748985552993 = 2.6987449903665475

Complexity:  10
Error:  4.483310381856224
Bias^2:  1.3066679565742672
Var:  3.176642425282004

4.483310381856224 >= 1.3066679565742672 +
3.176642425282004 = 4.483310381856271
```

```
Complexity:  11
Error:  5.946153881216655
Bias^2:  1.143281440703036
Var:  4.8028724405136165
5.946153881216655 >= 1.143281440703036 +
4.8028724405136165 = 5.946153881216652

Testing MSE for our k-folds CV:
[[21.986633786995704, 8.05137330205336, 6.011880845314531,
3.5193591191121376, 3.1630369417051973, 2.036100308138644,
2.175245447583792, 3.899015273872987, 2.0538778114820273,
5.477170903614403], [21.44681217106944, 7.879844591615398,
6.026223094892754, 3.480399437485483, 3.0879829142177466,
2.1411779426704145, 2.0376583980101453, 2.531580940161962,
3.3546199094647573, 2.261229381129488],
[21.394251554068642, 7.85106230572619, 6.257251209236729,
3.305789582034097, 3.1190682850755627, 1.9874046947082924,
2.13815847624519, 2.573934658856698, 3.16679291574127,
2.9959352597585314], [21.65293160926003,
7.979326107363779, 6.021691037746313, 3.440324902554833,
2.772739589954696, 2.027858156420131, 2.0851950503599808,
2.364269053920323, 2.070689324177793, 2.749138160485228],
[21.536973433453163, 7.968235178028359, 6.052360847617159,
3.243224961148909, 2.9808946082722203, 2.059966397469858,
1.9109018221728589, 2.4686275532855113,
2.9485029623859593, 3.091649936960191],
[21.14600905928912, 7.991308452122441, 6.147127621197541,
3.462463945506866, 3.029750087683468, 2.0136788352925956,
1.9176013474509048, 1.887330285010708, 1.8591367799082075,
2.1037950917202557]]

Best MSE for training:  0.6798353911154782
 in Poly_degree=10 and Parameter2=0
Best MSE for testing:  1.116780111855389
 in Poly_degree=10 and Parameter2=0
Best R2-score for training:  0.9987455410749885
 in Poly_degree=10 and Parameter2=0
Best R2-score for testing:  0.9979126742131896
 in Poly_degree=10 and Parameter2=0

Complexity:  2
Error:  17.250468099288973
Bias^2:  16.640950907109435
Var:  0.6095171921796788

17.250468099288973 >= 16.640950907109435 +
0.6095171921796788 = 17.250468099289115

Complexity:  3
Error:  7.381759200252668
Bias^2:  7.010318297486489
Var:  0.3714409027660959
```

```
7.381759200252668 >= 7.010318297486489 +
0.3714409027660959 = 7.381759200252585

Complexity:  4
Error:  5.753082695067827
Bias^2:  5.375073506881341
Var:  0.3780091881865065

5.753082695067827 >= 5.375073506881341 +
0.3780091881865065 = 5.753082695067848

Complexity:  5
Error:  3.4760095463697156
Bias^2:  3.203325510383217
Var:  0.2726840359865308

3.4760095463697156 >= 3.203325510383217 +
0.2726840359865308 = 3.476009546369748

Complexity:  6
Error:  2.896007029888318
Bias^2:  2.591400092577487
Var:  0.30460693731083494

2.896007029888318 >= 2.591400092577487 +
0.30460693731083494 = 2.896007029888322

Complexity:  7
Error:  2.4296967648895356
Bias^2:  2.114195798308914
Var:  0.3155009665806736

2.4296967648895356 >= 2.114195798308914 +
0.3155009665806736 = 2.429696764889588

Complexity:  8
Error:  2.091177052583386
Bias^2:  1.7940290435763242
Var:  0.29714800900704036

2.091177052583386 >= 1.7940290435763242 +
0.29714800900704036 = 2.0911770525833644

Complexity:  9
Error:  1.9218120622750676
Bias^2:  1.580508807856408
Var:  0.34130325441863096

1.9218120622750676 >= 1.580508807856408 +
0.34130325441863096 = 1.921812062275039

Complexity:  10
Error:  1.8966832193519092
Bias^2:  1.4990264698964997
```

```
Var:  0.39765674945547375

1.8966832193519092 >= 1.4990264698964997 +
0.39765674945547375 = 1.8966832193519734

Complexity:  11
Error:  1.8513635172297653
Bias^2:  1.4743530501747517
Var:  0.37701046705508706

1.8513635172297653 >= 1.4743530501747517 +
0.37701046705508706 = 1.8513635172298388

Testing MSE for our k-folds CV:
[[21.987603511957, 8.049301547051778, 6.002008691756702,
3.488257933255882, 3.154438490115589, 2.2612371367714212,
2.000278983327995, 1.7603429828378232, 1.795138047057224,
2.095590003123822], [21.4454774307964, 7.8781000950859665,
6.019430006542199, 3.459276593611875, 3.031322878099834,
2.1611154349794726, 1.898557904805533, 1.7670564010415386,
1.691929571323832, 1.8651860018298727],
[21.39443456293699, 7.850453589280515, 6.24755278495063,
3.292533102338268, 3.0326557996313364, 2.304095998171545,
2.00224718937014, 2.1355121387406575, 1.8002681244155423,
1.8291144900572291], [21.653249836379302,
7.978246467709998, 6.013472540799898, 3.435657574351909,
2.79604866194974, 2.298155879519525, 2.008553387839282,
2.061260980633402, 1.8193423776722641,
1.8037487849257197], [21.537344244407336,
7.96760227905754, 6.044720204136362, 3.239746982139076,
2.9788401554880277, 2.3753992975270224,
1.8953172691329674, 1.825302102077028, 1.8028497858145132,
1.8941648923299788], [21.14490050797267,
7.990686898121796, 6.139533424571152, 3.4414026288987456,
3.024252710396604, 2.2382337811885535, 1.9167029672365534,
1.7538819198471955, 1.7152461235047274,
1.836484435585049]]

Best MSE for training:  5.599576228202947
 in Poly_degree=11 and Parameter2=0
Best MSE for testing:  7.751923068448195
 in Poly_degree=11 and Parameter2=0.01
Best R2-score for training:  0.9896674423433208
 in Poly_degree=11 and Parameter2=0
Best R2-score for testing:  0.9855112132223951
 in Poly_degree=11 and Parameter2=0.01

Complexity:  2
Error:  20.276841487833195
Bias^2:  19.6125420068115
Var:  0.6642994810216318

20.276841487833195 >= 19.6125420068115 +
0.6642994810216318 = 20.27684148783313
```

```
Complexity:  3
Error:  13.481011252332006
Bias^2:  13.01597613465039
Var:  0.4650351176814038

13.481011252332006 >= 13.01597613465039 +
0.4650351176814038 = 13.481011252331795

Complexity:  4
Error:  10.006100568457436
Bias^2:  9.643019804655363
Var:  0.3630807638020801

10.006100568457436 >= 9.643019804655363 +
0.3630807638020801 = 10.006100568457443

Complexity:  5
Error:  8.928916033731987
Bias^2:  8.544411243806811
Var:  0.3845047899250052

8.928916033731987 >= 8.544411243806811 +
0.3845047899250052 = 8.928916033731817

Complexity:  6
Error:  9.170396390721008
Bias^2:  8.892136046032624
Var:  0.2782603446884047

9.170396390721008 >= 8.892136046032624 +
0.2782603446884047 = 9.170396390721029

Complexity:  7
Error:  9.806570520244321
Bias^2:  9.465568042238235
Var:  0.3410024780059168

9.806570520244321 >= 9.465568042238235 +
0.3410024780059168 = 9.806570520244152

Complexity:  8
Error:  9.68869185398975
Bias^2:  9.321869551026214
Var:  0.3668223029634368

9.68869185398975 >= 9.321869551026214 +
0.3668223029634368 = 9.688691853989651

Complexity:  9
Error:  9.509047542990338
Bias^2:  9.128962985638076
Var:  0.38008455735234825
```

```
9.509047542990338 >= 9.128962985638076 +
0.38008455735234825 = 9.509047542990423

Complexity:  10
Error:  8.863237309441741
Bias^2:  8.498210172982535
Var:  0.3650271364591484

8.863237309441741 >= 8.498210172982535 +
0.3650271364591484 = 8.863237309441683

Complexity:  11
Error:  8.573599449764004
Bias^2:  8.233484529110314
Var:  0.3401149206535745

8.573599449764004 >= 8.233484529110314 +
0.3401149206535745 = 8.573599449763888

Testing MSE for our k-folds CV:
[[24.695679113661733, 14.876697223651842,
9.77467690745872, 8.902491859152269, 8.291697152879511,
9.0405229216526, 7.6103392381502175, 7.089681033795401,
6.502590829203402, 6.334401171034076],
[25.155246458272035, 14.269911160769501,
10.032805980758987, 8.267906265978313, 8.313261924655297,
9.150354488660447, 7.902771922604605, 7.062936190435219,
6.523717542483937, 6.16581822071175], [24.296911503563614,
14.298084963731332, 10.141208927139811, 8.436947941481334,
8.234969147441525, 8.984652852788612, 7.901514078889897,
7.064660959892436, 6.943400869479371, 6.4305958062917],
[24.628998687905842, 14.37214835461819, 9.963227425819992,
8.44309472269075, 8.54055432841713, 9.151798475844878,
7.978089498027972, 7.223240658395934, 6.77923986674886,
6.1471852795938], [24.585683869517474, 14.463196193922288,
9.825401950051228, 8.442307774946627, 8.271344829639096,
8.471923284434155, 7.835216915666687, 7.089542303682218,
6.744802194676784, 6.1610782808480415],
[24.66710843907483, 14.81185772458505, 9.655917823016413,
8.370317160207879, 8.51432141630251, 8.734933797456934,
7.857850793477672, 7.269980589228931, 6.487743475333372,
6.109067721337664]]
```