Regression Analysis and Resampling Methods

G. Durante, P. A. Lopez Torres, E. Ottoboni

October 7 2024

Abstract

We evaluated various approaches to fitting linear regression models to the Franke function, focusing on Ordinary Least Squares (OLS), Ridge, and Lasso regression techniques. To rigorously assess these methods, we employed resampling techniques, including bootstrapping and cross-validation, to ensure a thorough evaluation of model performance. Beyond the practical analysis, we also examined the theoretical foundations of these models, with particular attention to their distributional properties and the bias-variance trade-off. This provided a deeper understanding of the strengths and limitations of each approach in the context of our specific application.

1. INTRODUCTION

Linear algebra provides a framework for solving systems of equations through matrix inversion. More generally, any relationship between independent variables x_i and dependent variables y_i can be represented as a set of equations. Although these equations may not always have an exact solution, especially when random noise is present, nevertheless solutions can be derived using linear algebra techniques. By collecting real-world data, assigning values to the input parameters, and relating them to output parameters through a system of equations, we can quantify their relationship. This measure can then be leveraged to predict outputs for new input data.

In this report, we focus on linear regression, which can be interpreted as a solution to a system of linear equations. We focus on the performance of Ordinary Least Squares (OLS), Ridge, and Lasso regression techniques. Our goal is to understand which methods perform best under varying polynomial degrees on the Franke function with the addition of an added stochastic noise with normal distribution $N(0, 1)$.

Initially, we fit a real-valued multivariate function $\mathbb{R}^2$ to a polynomial model incorporating two independent variables. The cross-validation method is used to partition the dataset into multiple training and test sets. After training the models, we evaluate their performance using the mean squared error and the R^2 coefficient, particularly assessing the effect of different values of the hyperparameter λ for Ridge and Lasso. We also use the bootstrap technique as an alternative resampling method to analyze the biases and variances in the predictions.

Additionally, we explore a real-world application by using a terrain dataset from Kartverket (Norway). Here, the independent variable is the height, and the dependent variables become the x- and y-coordinates. We again analyze the data using polynomials of different degrees, and apply the different models.

2. THEORY

To fit a polynomial of degree p to a set of data points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, one must solve the following system of equations:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots + \beta_p x_1^p, \\ y_2 &= \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + \cdots + \beta_p x_2^p, \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_n + \beta_2 x_n^2 + \cdots + \beta_p x_n^p. \end{aligned} \tag{1}$$

The above system can be rewritten in matrix form as:

$$\tilde{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}, \tag{2}$$

Where $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\hat{\boldsymbol{\beta}} \in \mathbb{R}^p$, and $\hat{\mathbf{y}} \in \mathbb{R}^n$. It is not guaranteed that the system in (1) is consistent, which implies that an exact solution may not exist. Therefore, we must define a cost function $C(\hat{\boldsymbol{\beta}})$, and find the $\hat{\boldsymbol{\beta}}$ coefficients that minimize it, in order to achieve the best fit for the data. The design of the cost function varies depending on the type of regression used. We will now describe the behavior of this function in different linear regression methods.

Ordinary Least Squares (OLS)

In the context of Ordinary Least Squares (OLS), the cost function is defined as the Mean Squared Error (MSE):

$$C(\mathbf{X}, \hat{\boldsymbol{\beta}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 = \frac{1}{n} \left\| \hat{\mathbf{y}} - \mathbf{X}\hat{\boldsymbol{\beta}} \right\|_2^2, \tag{3}$$

Where $\tilde{y}_i$ are the predicted values obtained from $\tilde{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$. To determine the optimal $\hat{\boldsymbol{\beta}}$ coefficients,

we differentiate the cost function with respect to $\hat{\boldsymbol{\beta}}$ and set it to zero. The resulting solution is given by:

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

The variance of the β_i coefficient, obtained using Ordinary Least Squares (OLS):

$$\text{var}(\beta_i) = \left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right)_{ii}\sigma^2, \tag{5}$$

For the derivation of the expectation values for OLS and β parameters, check Appendix A and Appendix B.

Ridge

For Ridge regression, the cost function is defined as:

$$C(\mathbf{X}, \hat{\boldsymbol{\beta}}) = \left\|\hat{\mathbf{y}} - \mathbf{X}\hat{\boldsymbol{\beta}}\right\|_2^2 + \lambda\left\|\hat{\boldsymbol{\beta}}\right\|_2^2, \tag{6}$$

where $\tilde{y}_i$ represents the same predicted values as defined in Equation (2). In this case, a regularization parameter $\lambda \geq 0$ is introduced. As λ increases, the regularization term $\lambda\left\|\hat{\boldsymbol{\beta}}\right\|_2^2$ begins to dominate, leading to smaller values of the $\hat{\boldsymbol{\beta}}$ coefficients. Thus, the parameter λ effectively shrinks the coefficients of $\hat{\boldsymbol{\beta}}$. This introduces a bias to the estimation, meaning that the expectation of the prediction is not the same as in Ordinary Least Squares (OLS), which minimizes the Mean Squared Error (MSE).

Furthermore, for each coefficient, the variance is defined as:

$$\begin{aligned}\text{var}(\beta_i) = \sigma^2 \Big[&\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \\ &\cdot \mathbf{X}^T\mathbf{X} \cdot \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\Big]_{ii},\end{aligned} \tag{7}$$

that can be rewritten in a more compact way as:

$$\begin{aligned}\text{var}(\beta_i) &= \sigma^2\left[\mathbf{A}\mathbf{X}^T\mathbf{X}\mathbf{A}\right]_{ii}, \\ &\text{where } \mathbf{A} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}.\end{aligned} \tag{8}$$

Lasso

The final method for linear regression that we will examine is the Least Absolute Shrinkage and Selection Operator (Lasso). The cost function is defined as:

$$C(\mathbf{X}, \hat{\boldsymbol{\beta}}) = \frac{1}{2}\left\|\hat{\mathbf{y}} - \mathbf{X}\hat{\boldsymbol{\beta}}\right\|_2^2 + \lambda\left\|\hat{\boldsymbol{\beta}}\right\|_1, \tag{9}$$

where the minimization problem does not have a closed-form solution. Therefore, to optimize $\hat{\boldsymbol{\beta}}$, one must employ numerical approaches, such as gradient descent. Similar to Ridge regression, Lasso introduces a bias to the estimation process.

Bias-Variance Tradeoff

When modeling, the goal is to approximate the test data while minimizing the Mean Squared Error (MSE). Increasing the complexity of the model can enhance its accuracy in reducing the error in the training data; however, a higher complexity may also lead to overfitting, resulting in a greater error in the test data. This dual effect of increasing a model's complexity can be illustrated by rewriting the expression for the MSE:

$$\textbf{MSE} = \frac{1}{n}\sum_i\left(\hat{f}_i - \mathbb{E}[\tilde{y}]\right)^2 + \frac{1}{n}\sum_i\left(\hat{\tilde{y}}_i - \mathbb{E}[\tilde{y}]\right)^2 + \sigma^2. \tag{10}$$

The first term is referred to as the bias, which is the squared distance between the expected value of the model and the true function. As we increase the complexity of the model, this term should decrease, allowing for more flexibility in approximating the actual function.

The second term in Equation (10) represents the variance, which quantifies the squared distance between the estimated values and their expected values. This term typically increases with model complexity, stemming from the fact that we have a finite number of data points. As complexity rises, the fit becomes heavily reliant on the specific dataset, which is less problematic at lower complexities, where the model has fewer degrees of freedom. However, at higher levels of complexity, the model may begin to exploit the noise in the data, leading to greater variance.

The derivation can be found in the Appendix C. Based on our assumption, excessive model complexity leads to overfitting, where the model is too dependent on the specific data, causing the true MSE to increase. Conversely, overly simplistic models fail to represent the actual function, leading to underfitting and similarly high MSE. This situation creates a tradeoff, where an optimal model complexity must be found between these two extremes. This tradeoff is known as the bias-variance tradeoff.

Calculations

The proposed models integrates the values along the x and y-axes into a n-th order polynomial, incorporating all cross terms. Each polynomial term corresponds to a column in the design matrix, constructed from the evaluated values at various data points. The complexity of the model is contingent upon the degree of the polynomial employed for fitting, enabling a comparative analysis of different linear regression methodologies across different levels of complexity.

Additionally, we explore the influence of noise and the number of data points. For the Ridge and Lasso regression techniques, which optimize the cost

functions defined in Equations (6) and (9), we also examine the impact of the λ hyperparameter.

In the OLS, the computation of the inverse of the matrix $\mathbf{X}^T\mathbf{X}$ can be computationally intensive and susceptible to numerical inaccuracies. To construct and design the feature matrix efficiently, we employ the `PolynomialFeatures` function from the `scikit-learn` preprocessing module. Afterwards, we are able to compute the $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ parameters by integrating the created design matrix into the cost function minimization formula.

When it comes to Ridge regression, the same method applies. However, it is necessary to modify the $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ formula by adequately adding the λ parameters.

For the Lasso method, there is no closed-form solution for $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$, so we avoid implementing custom code to estimate the coefficients. Instead, we use the `sklearn.linear_model.Lasso` class from the `scikit-learn` library. This class solves the problem using an iterative approach, where we can define a tolerance and a maximum number of iterations. The algorithm employs coordinate descent, updating one coefficient at a time by computing the partial derivative of the cost function with respect to the coefficient, and then modifying it based on the derivative and a step size, or learning rate. This learning rate is automatically determined by `scikit-learn`.

The primary goal is to explore how the complexity of the model, controlled by the degree of the polynomial, affects the accuracy and generalizability of the model.

After combining the independent variables into a feature matrix $\mathbf{X}$, we split the data into training and test sets, using 80% of the available data for training and 20% for testing.

We fit a polynomial regression model by iterating over polynomial degrees from 1 to 15. For each degree d, polynomial features are generated for both training and test data, and a linear regression model is trained on the transformed data.

$$\text{PolyFeatures}(d) : \mathbf{X} \to \mathbf{X}^d_{\text{poly}}$$

$$\text{Model: } \hat{z}_{\text{train}} = f(\mathbf{X}_{\text{train}}), \hat{z}_{\text{test}} = f(\mathbf{X}_{\text{test}})$$

For each model, we compute the Mean Squared Error (MSE) and R^2 score to assess their performance.

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{z}_i)^2 \tag{11}$$

$$R^2 = 1 - \frac{\sum (z_i - \hat{z}_i)^2}{\sum (z_i - \bar{z})^2} \tag{12}$$

The performance metrics, including MSE and R^2, are recorded for both the training and test sets. This allows us to track how the model performs as its complexity increases. Finally, we observe how the MSE and R^2 values for both the training and test sets change with the degree of the polynomial.

By analyzing the performance of the polynomial regression models at various degrees, we can determine the optimal degree that balances the trade-off between bias and variance. Models with too low a degree may underfit, while those with too high a degree may overfit the data.

In addition, both in Ridge and Lasso regression we examine the impact of regularization on the model performance. As we have seen in Equation (6) and (9), α is the regularization parameter that controls the penalty applied to the β_j model coefficients. The regularization helps prevent overfitting by shrinking the coefficients, effectively reducing the model complexity. We evaluate the performance of both Ridge and Lasso regression for multiple values of α and select the best parameter based on the minimum MSE calculated on the test set:

$$\alpha_{\text{optimal}} = \arg\min_{\alpha} \mathbf{MSE}_{\text{test}}$$

The optimal value of α is reported along with the corresponding minimum MSE. To complement the Lasso regression, in ridge regression we have α that is the regularization parameter that controls the penalty applied to the squared coefficients β_j. Unlike Lasso, Ridge does not shrink coefficients to zero but reduces their magnitude proportionally, making it ideal for handling multicollinearity and high-dimensional feature spaces. The value of α is varied systematically to determine its effect on model performance. For each α, the Mean Squared Error (MSE) on the test set is evaluated, and the optimal α is selected based on the minimum MSE:

$$\alpha_{\text{optimal}} = \arg\min_{\alpha} \mathbf{MSE}_{\text{test}}$$

The optimal value of α is reported along with the corresponding minimum MSE.

Dataset

To evaluate the performance of various linear regression methods, we utilize two distinct datasets. Initially, we examine the Franke function, defined as:

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned} \tag{13}$$

To analyze the characteristics of the Franke function, we first created a contour plot of the data, as shown in Figure **1**. This visualization illustrates the function's complex shape, highlighting potential challenges associated with fitting a polynomial

model. The contour plot reveals two distinct maxima, one minimum, and two saddle points within the region where the data is generated. Such features indicate that simple polynomial fitting may struggle to accurately capture the function's behavior, particularly in areas with rapid changes in elevation. This complexity underscores the need for careful model selection and validation when employing polynomial regression techniques for approximation in this context.

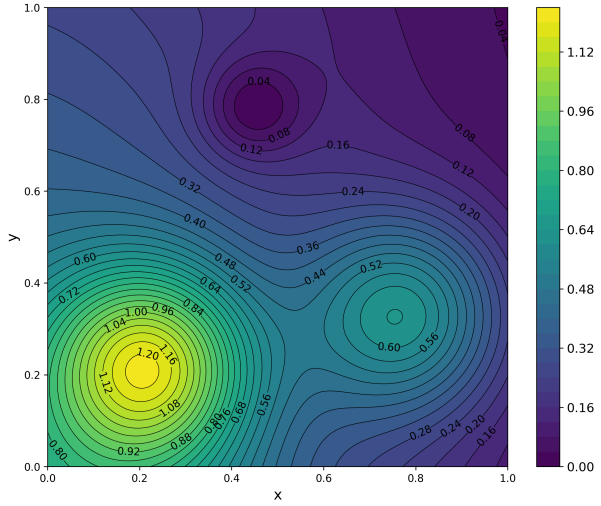

Figure 1: Contour Plot of Franke Function with 60 000 datapoints.

Our analysis includes evaluating this function with and without the addition of Gaussian noise. To generate a noisy observation z, we utilize the following expression:

$$z = f(x, y) + \epsilon, \tag{14}$$

where ϵ represents the noise term defined as:

$$\epsilon = 0.1 \cdot \mathrm{N}(0, 1), \tag{15}$$

Here, $\mathrm{N}(0, 1)$ indicates a random variable drawn from a normal distribution with mean 0 and standard deviation 1. The factor 0.1 scales the noise to a desired level, simulating realistic measurement conditions.

We also examine a dataset comprising height data from a region in Norway, referred to as *"SRTM data Norway 1-.tif"* (Figure **15**). This dataset contains $3,601 \times 1,801$ data points, for a total of $6,485,401$, making its size quite significant. To facilitate our analysis, we focus on a smaller subset, randomly sampling a NxN pixel area.

Given that we are analyzing a two-dimensional surface, we cannot construct the X-matrix as derived from Equation 4. Instead, we will use the `NumPy` function `np.column_stack` along with `PolynomialFeatures` to create the matrix $\mathbf{X}$. This function generates the matrix for a polynomial of degree (i, j), allowing for different polynomial degrees across the two dimensions. However, we will utilize the same degree for both dimensions and refer to the degree of the fit with a single number.

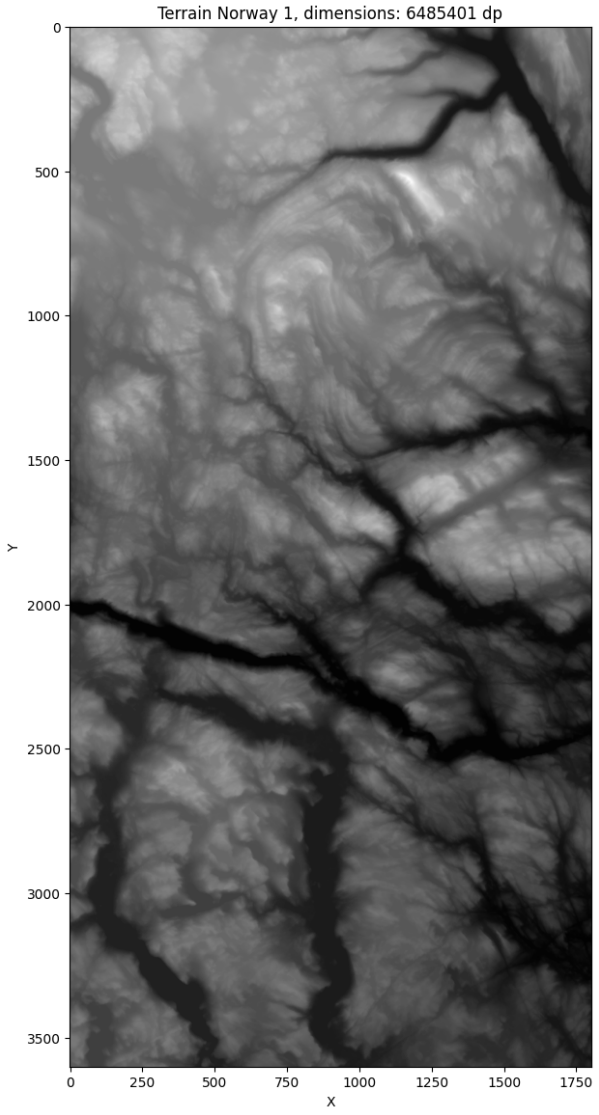

Figure 2: Terrain from Norway, full dataset.

Data Preprocessing

Data preprocessing is a critical step in improving model performance, and common techniques include mean centering and standardization. While mean centering ensures that features are centered around zero, standardization scales the data to have a mean of zero and a standard deviation of one.

In our case, $\mathbf{X} \in \mathbb{R}^{n \times m}$ represents the training dataset with n samples and m features. The mean j of each feature is calculated as:

$$\bar{x_j} = \frac{1}{n} \sum_{i=1}^{n} x_{ij}, \quad j = 1, 2, \ldots, m. \tag{16}$$

The mean-centered dataset X_{cent} is then given

by:

$$\mathbf{X}_{\text{cent}} = \mathbf{X} - \bar{\mathbf{X}}. \tag{17}$$

Standardization is performed using `Standard Scaler` from `Scikit-learn`, which standardizes the features by removing the mean and scaling to unit variance. The standardized dataset X_{st} is computed as:

$$\mathbf{X}_{\text{st}} = \frac{\mathbf{X} - \mu}{\sigma}, \tag{18}$$

where μ and σ are the mean and standard deviation of the feature, respectively.

The mean and standard deviation are calculated during the fitting process:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}. \tag{19}$$

The `fit_transform` method computes the mean and standard deviation from the training data and applies the transformation, while the `transform` method applies the same transformation to the test data using the parameters computed from the training set. We do not fit again the test data to avoid any possible data leakage.

By centering and standardizing the features, we could improve the performance and reliability of our models. Normalization allows data to be well-prepared for training algorithms that are sensitive to feature scales and distributions. For each model analyzed, a discussion regarding the usefulness or not of scaling data takes place.

Performance

The main features we employ to analyze the performance of our models are the Mean Squared Error (MSE) and the Coefficient of Determination (R^2), introduced in Equation 11 and Equation 12, respectevely. The MSE is a commonly used cost function, minimized by ordinary least squares (OLS) and regression-based models, while the R^2 score quantifies the proportion of variance in the dependent variable explained by the model.

To rigorously evaluate the OLS model and the regularized techniques like Ridge and Lasso, the dataset is initially partitioned into training and testing subsets. The training data is used to fit the model, while the testing data is utilized to compute the MSE and R^2 values, thus preventing overfitting. For an more robust evaluation, we employ the k-fold cross-validation technique, where the data is divided into k subsets, using one subset for testing and the remaining $k-1$ subsets for training. This process is repeated k times, and the average of the performance metrics (MSE and/or R^2) is taken as a measure of model performance.

For models like Ridge and Lasso, cross-validation is essential to select the optimal hyperparameter λ. By iterating over a range of λ values, the model is trained on multiple folds, and the average cross-validation error is minimized to find the best λ. Once the best hyperparameter is determined, the final model is retrained on both the training and validation sets to refine the parameter estimates, and its performance is then evaluated on the unseen test set.

Additionally, the bootstrapping method is employed. This involves resampling the data with replacement to generate multiple datasets. This approach allows us to estimate the variance and bias of the model's predictions. The combined use of cross-validation and bootstrapping enables a comprehensive analysis of the model's predictive accuracy, especially when applied to the Franke function using OLS, Ridge, and Lasso regression techniques, under varying noise levels and hyperparameters.

In summary, through all these strategies, we ensure robust hyperparameter tuning, accurate performance evaluation, and a deeper understanding of the model's generalization capabilities.

3. RESULTS

Ordinary Least Squares

First, we fit the Franke function with 1 000 data points and run an OLS regression. The polynomial function allows us to fit linear regression models to non-linear data by introducing higher order terms. The maximum polynomial degree is set to 15. For each polynomial degree, and for both the test and training datasets, we compute the MSE and R^2. Two plots are then generated: one for the MSE and another for R^2 over different polynomial degrees, facilitating the analysis of trends in model performance. The results are shown in Figure **3** and the values are stored in Table **1**. The motivation behind this process is to explore the relationship between polynomial degree and model performance. A model that is too simple may fail to capture data complexity (underfitting), while a model that is too complex may learn noise instead of the underlying pattern (overfitting). By evaluating MSE and R^2 across various polynomial degrees, we can identify the optimal model complexity that balances bias and variance, leading to improved predictive accuracy.

In Figure **3**, the analysis of training data for both the MSE (blue solid line) and R^2 (red solid line) indicates an improvement of the model as the degree of the polynomial increases. This is clearly shown by the sharp decrease in the MSE and the R^2 rise. This behavior suggests that more complex models (higher polynomial degrees) fit the training data better, reducing errors and increasing the proportion of variance explained by the model.

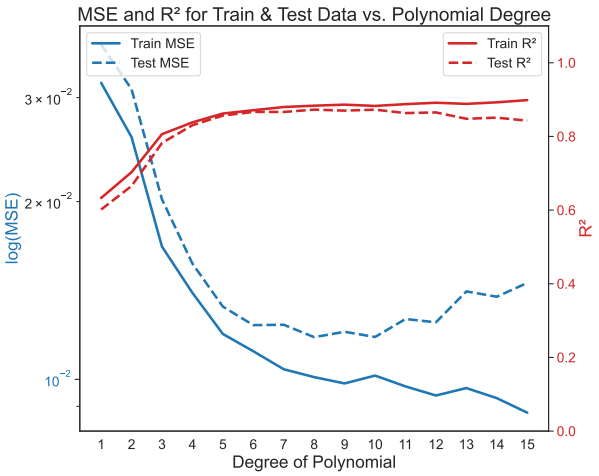

Figure 3: MSE and R^2 for Train and Test Data vs. Polynomial Degree (1 000 observations). The values on the y-axis scale logarithmically. This plot illustrates how increasing the polynomial degree impacts both the error and the predictive capability of the model.

When it comes to test data the situation is different. Initially, as the degree of the polynomial rises up, the MSE (blue dashed line) decreases, while R^2 (red dashed line) increases. However, beyond a certain point identifiable with polynomial degree 5, the test MSE begins to increase, and the test R^2 stabilizes and even starts to drop. This is a clear indication of overfitting: while the model fits the training data very well for high degrees, it generalizes poorly to unseen test data, suggesting that the added complexity captures noise rather than underlying patterns in the data.

D	Train MSE	Test MSE	Train R^2	Test R^2
1	0.0318	0.0369	0.6329	0.6015
2	0.0257	0.0310	0.7030	0.6655
3	0.0168	0.0202	0.8062	0.7823
4	0.0140	0.0157	0.8382	0.8305
5	0.0119	0.0133	0.8621	0.8567
6	0.0112	0.0124	0.8711	0.8666
7	0.0104	0.0124	0.8798	0.8664
8	0.0101	0.0118	0.8835	0.8727
9	0.0098	0.0120	0.8863	0.8700
10	0.0101	0.0118	0.8828	0.8727
11	0.0097	0.0127	0.8876	0.8634
12	0.0094	0.0125	0.8915	0.8652
13	0.0097	0.0141	0.8883	0.8479
14	0.0093	0.0138	0.8926	0.8510
15	0.0088	0.0146	0.8986	0.8428

Table 1: R^2 and MSE Values for Different Polynomial Degrees for OLS(1.000 observations).

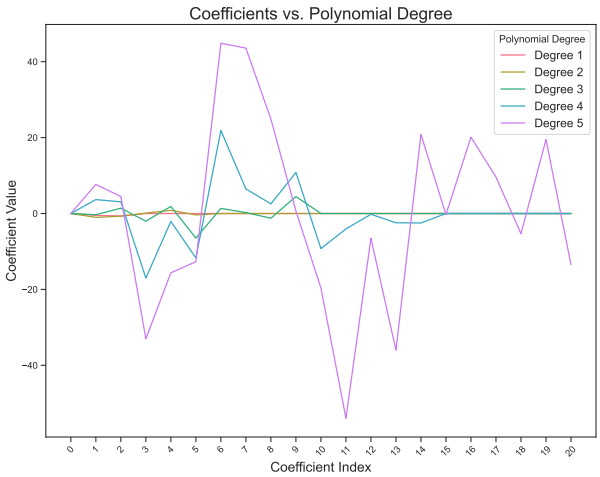

Figure 4: Coefficient trends (1 000 observations) for Ordinary Least Squares models fitted to polynomial features of varying degrees (from 1 to 5).

In Figure **4** it is possible to appreciate coefficient trends for Ordinary Least Squares (OLS) models fitted to polynomial features of varying degree (from 1 to 5). For models with a low polynomial degree, such as degree 1 or 2, the coefficient values remain relatively stable and close to zero. This stability is an expected outcome, as low-degree models are less flexible and unable to capture complex patterns in the data. They produce smoother fitting curves with minimal variance in the coefficient values. For models with intermediate polynomial degrees, such as 3 or 4, the coefficient values start to show greater fluctuations. While still controlled, the coefficients begin to diverge more, reflecting a higher sensitivity to variations in the data. This divergence indicates a greater risk of overfitting, as the model becomes more complex and tries to fit more intricate patterns within the dataset. Finally, the model with the highest polynomial degree (degree 5) exhibits significant variance and high-magnitude coefficient values. This behavior suggests that the model is starting to capture noise in the data, a clear indication of overfitting. As a result, the coefficients become increasingly erratic, and the overall model complexity rises sharply without a corresponding improvement in the model's predictive performance.

In Figure **5** a comparison between the OLS model and its scaled variant is shown. In unscaled models, polynomial features can cause large variations due to the dominance of higher-order terms, resulting in significant numerical instability and potential overfitting.

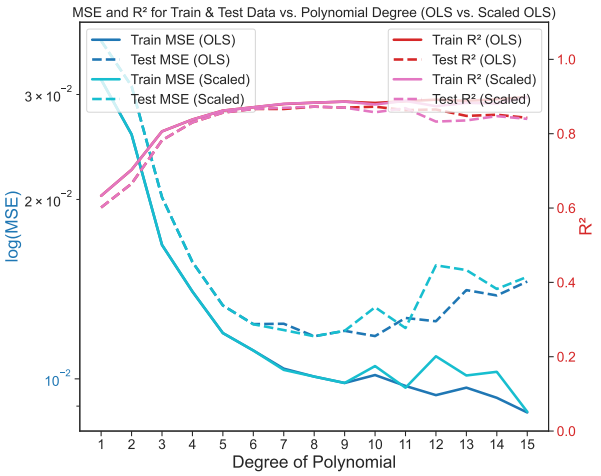

Figure 5: The graph compares the Mean Squared Error (MSE) and R^2 values for both Ordinary Least Squares and Scaled OLS models across polynomial degrees 1 to 15 (1 000 observations).

The R^2 values, remain relatively consistent between the scaled and unscaled models, especially for lower polynomial degrees, indicating similar performance in explaining the variance. However, as the polynomial degree increases, the unscaled model exhibits more erratic behavior and overfitting, as evidenced by the increased variability in test R^2. In contrast, the scaled model shows smoother convergence, suggesting better generalization properties. Data scaling is advantageous in polynomial regression because it mitigates the issue of exploding or diminishing values for higher-degree terms. This ensures a more stable optimization process and better numerical properties, leading to a more robust fit.

Ridge

When it comes to Ridge regression, the analysis performed follows the same steps as the OLS one. At the same time, the introduction of a λ parameter in the cost function (See equation 6) provides an additional degree of difficulty to consider.

We start by analyzing the unscaled data. As previously done, we compute and plot the change in the MSE and R^2 values as the polynomial degree increases. For a more coherent analysis and to compare the obtained results with the OLS ones, the impact of noise is kept at the same minimum level as in previous considerations (See Equation 15). The difference is that in this case, before we can calculate the actual MSE and R^2 values for each polynomial degree, we have to identify the corresponding optimal λ parameter. This is done by applying the `GridSearchCv` class from the `scikit-learn` library [1]. This allows us to use both cross-validation and grid search techniques for hyperparameter tuning [2]. Specifically, using cross-validation, we divide the training set into k-folds (5 in our case), and again one of these k-folds represents the validation set. We then compute the error for each of these different validation sets. The resulting average error tells us how well each lambda parameter behaved. This procedure is performed for each polynomial degree considered. Finally, the grid search allows us to apply the cross-validation technique for each value of lambda (in our case varying from a minimum of 10^{-10} to a maximum of 10^{10}). By confronting the different average errors obtained, we can then choose the best lambda parameter for each polynomial degree.

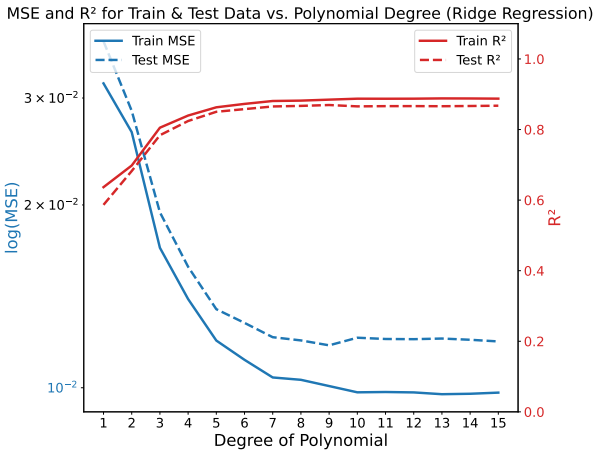

Figure 6: MSE and R^2 for Train and Test Data vs. Polynomial Degree (1 000 observations) for Ridge regression. The regularization of the β parameters can be appreciated.

The above Figure **6** gives us many interesting insights. Similarly to the OLS case, the overall performance of the model improves as the polynomial degree increases. This is clearly seen by the decrease in the MSE values and the increase in the R^2 values. However, unlike the latter case, here the regularization of the β parameters takes place, avoiding overfitting for higher polynomial degrees. In fact, it is possible to see how the MSE values related to the test data do not *'explode'* for higher degrees. At the same time, the R^2 values remain high and stable. The best performance of the model can be observed around degree 9, similar to what we already observed for the OLS regression. However, the overall performance of the Ridge regression with 1 000 data points is slightly better than the OLS one.

In summary, the performance of the Ridge regression model improves with higher polynomial degrees, with a peak around degree 9. With 1 000 data points, the overall performance is slightly improved and more stable compared to the OLS regression, mainly due to the regularization factor. Table **2** shows in more detail the different MSE and R^2 values for each polynomial degree.

D	Train MSE	Test MSE	Train R^2	Test R^2
1	0.0317	0.0372	0.6367	0.5866
2	0.0264	0.0286	0.6980	0.6824
3	0.0170	0.0194	0.8052	0.7838
4	0.0140	0.0158	0.8396	0.8242
5	0.0120	0.0135	0.8630	0.8503
6	0.0111	0.0128	0.8726	0.8579
7	0.0104	0.0121	0.8809	0.8654
8	0.0103	0.0120	0.8819	0.8670
9	0.0101	0.0117	0.8847	0.8695
10	0.0098	0.0121	0.8874	0.8656
11	0.0098	0.0120	0.8873	0.8663
12	0.0098	0.0120	0.8874	0.8664
13	0.0098	0.0121	0.8882	0.8661
14	0.0098	0.0120	0.8880	0.8667
15	0.0098	0.0119	0.8875	0.8675

Table 2: R^2 and MSE Values for Different Polynomial Degrees (1 000 observations) for Ridge regression.

Now it is interesting to analyze the impact of the different λ parameters on the model performance. In particular, we plot the different λ values at the various polynomial degrees, in order to discover any underlying trend.

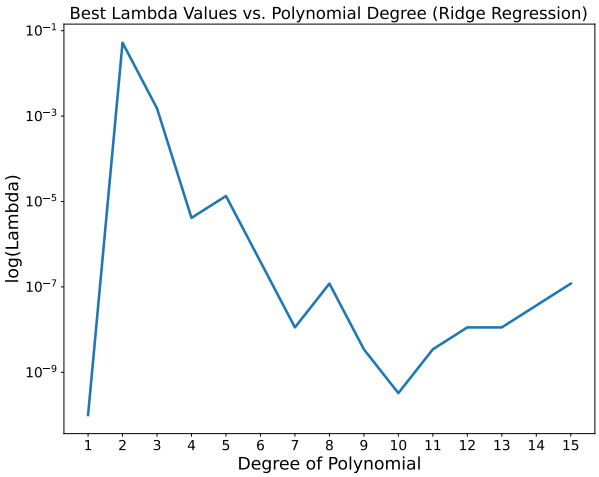

Figure 7: Best Lambda Values vs. Polynomial Degree (1 000 observations) for Ridge regression.

The above Figure **7** shows an interesting behavior of the regularization parameters. As the polynomial degree increases, there is a reduction in the λ values, indicating that less bias is introduced into the model. This makes sense because as the degree of the polynomial grows, the OLS model tends to perform better, requiring less regularization. However, after a certain point, the λ values start to rise again to counteract the overfitting problem, which typically affects OLS regression models at higher degrees. This increase in regularization helps explain why, in the MSE graph, the error levels appear more controlled for the Ridge regression.

The above analysis only takes into account the best lambda values for each polynomial degree. It could also be interesting to analyze how the overall model performance changes with different lambda values (not exclusively the best ones) and different polynomial degrees. Figure **8** gives us a visual representation of this analysis.

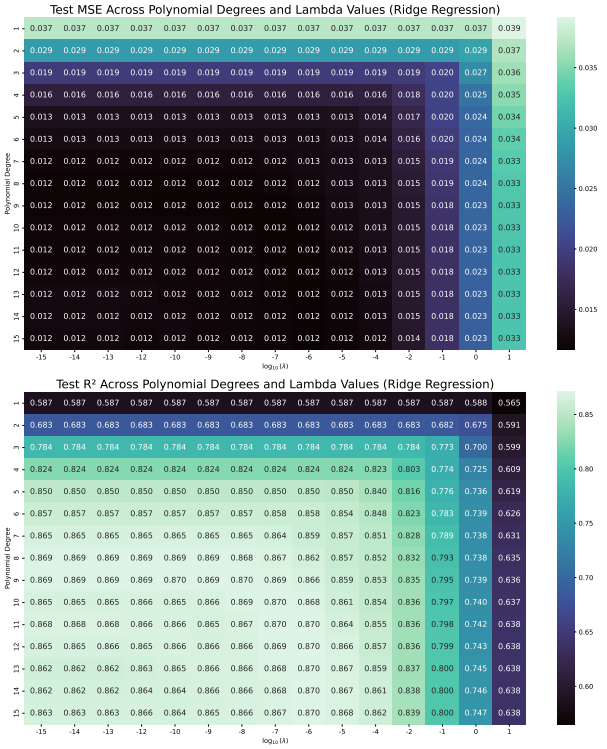

Figure 8: Test MSE and R^2 for different polynomial degrees (from 1 to 15) and lambda values (15 values ranging from 10^{-15} to 10^{1}), for a total of 1 000 observations.

The lower left cells show the best results for both the MSE and R^2, while the upper right cells show the worst results. There is a clear pattern in both heatmaps: higher polynomial degrees and lower lambda values lead to the best results. This is consistent with what we have seen so far and gives us a broader context on how to choose the more appropriate Ridge regression parameters.

So far, our analysis has focused on unscaled data. In order to get a more comprehensive and coherent view of the problem at hand, we now scale our data and then evaluate the performance of the model, as we have done in the previous OLS analysis. In particular, we normalize our data by using the `StandardScaler` class belonging to the `scikit-learn` library [1]. The scaling is applied to both the dependent and independent variables. However, this analysis does not show much improvement in terms of error. The MSE values are considerably higher than before, while the R^2 values behave similarly. For this reason, the main results of this Ridge regression analysis are given with respect to the unscaled data.

Lasso

The same analysis conducted for the previous methods has been performed using Lasso Regression. Similar to the case of Ridge Regression, 20 values of λ were considered within the range $[10^{-5}, 10^{5}]$, chosen using the logspace operator.

After fitting the Franke function to 1 000 data points and performing the Lasso regression, the data were split into an 80 per cent training set, a 10 per cent validation set and a 10 per cent test set for each polynomial degree. This approach allowed an initial analysis using the training and validation sets to determine the optimal parameter α, specifically the one that produced the lowest mean square error (MSE). The selected model was then applied to the test set.

This procedure was repeated for each polynomial degree, after which the values of MSE and R^2 were calculated. To facilitate understanding of the model's performance, a graph was generated showing the MSE and R^2 data for different polynomial degrees. The results are shown in Figure **9**, and it can be seen that the scaled data in the Lasso case show a higher MSE and a lower R^2 than the unscaled values.

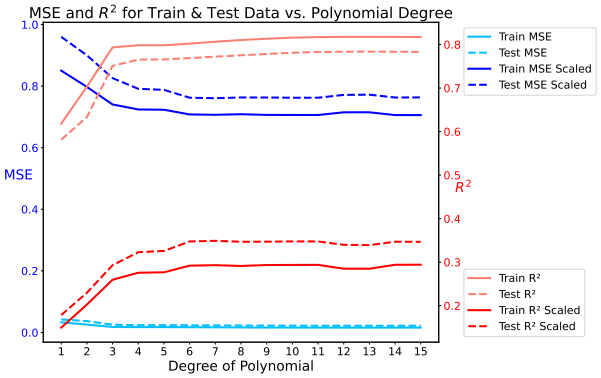

Figure 9: MSE and R^2 for Train and Test Data vs. Polynomial Degree (1 000 observations). The y-axis values are scaled logarithmically.

The decrease in R^2 could be attributed to several factors. Primarily, certain models may be sensitive to scaling, especially if the features are not standardized (or normalized) concerning their variability. Additionally, it could be concluded that this reduction is due to the Lasso introducing a penalty α to the cost function, which has the effect of shrinking the coefficients of some features to zero. This leads to a simpler model, helping to prevent overfitting, but it may also diminish the model's ability to fit the data, particularly if the penalty is too high.

Despite this, to maintain symmetry and achieve a more comprehensive comparison with the previously used models, we will utilize the scaled data. In this case, our model's test data MSE for optimal parameters $\lambda = 0.0003801894$ with polynomial degree $= 7$ denotes at MSE $= 0.7608$.

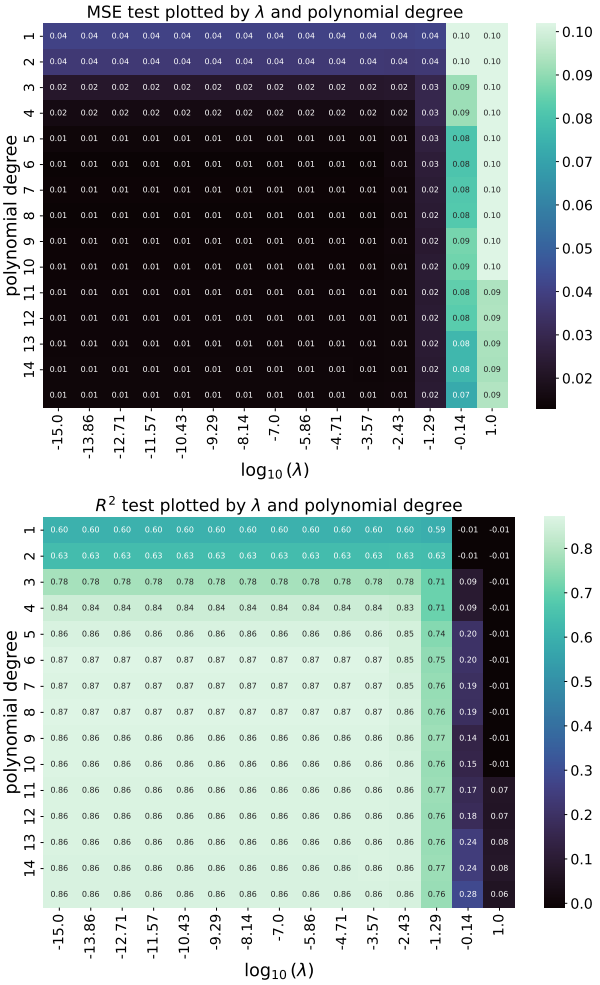

Figure 10: Test MSE and R^2 for different polynomial degrees (from 1 to 15) and lambda values (20 values ranging from 10^{-5} to 10^{5}), for a total of 1 000 observations.

From the analysis of the results, the OLS model proved to be the most suitable for the dataset studied, due to a combination of solid predictive performance and better interpretability than the Ridge and Lasso models. The slight reduction in variance achieved by Ridge did not justify the loss of transparency of the model, while Lasso, although interesting in terms of variable selection, sacrificed some predictive accuracy. Therefore, considering the absence of critical multicollinearity and the robustness of the performance, OLS represents the best compromise for this specific application. As we can seen in the Figure **11**

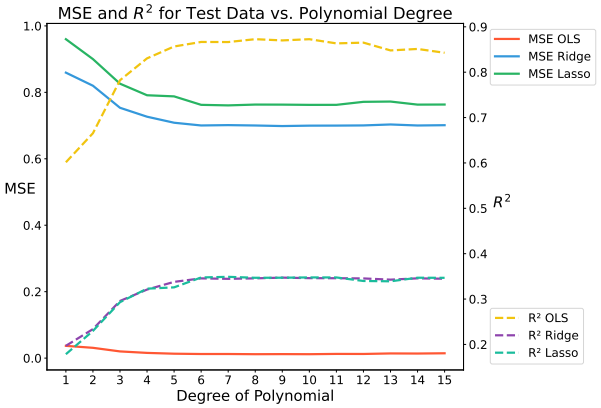

Figure 11: Performance comparison of the three methods with MSE and R^2.

Bootstrap

Building on the exploration of OLS regression, the next step involves a more in-depth analysis of how model complexity affects bias, variance and overall error. To this end, we use bootstrap sampling, a resampling technique that allows us to estimate the variability of these components more robustly. Specifically, we use 100 bootstrap samples for our analysis.

We start by generating the data and setting the number of bootstrap samples to 100. To visualize this, we compute the MSE for different polynomial degrees and plot the results, with the MSE on the y-axis and the polynomial degree on the x-axis. For clarity, we use a logarithmic scale (base 10). These data are displayed in the Figure **12**.

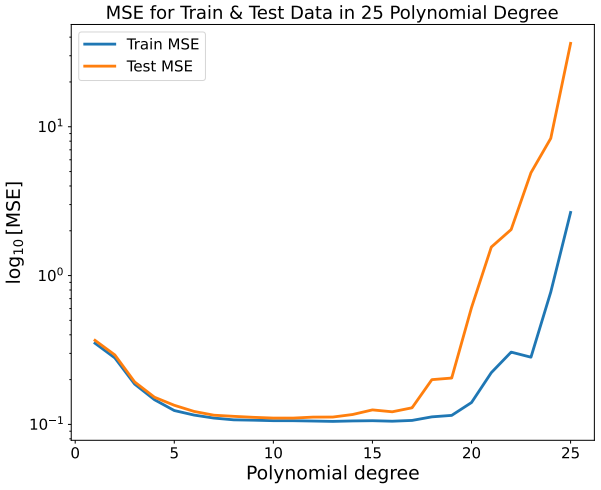

Figure 12: We can appreciate the U-shape of the test MSE and how it is less regular compared to the train one.

For the first four polynomial degrees, the training and test MSE curves overlap, indicating high bias and low variance. This is a case of underfitting, where the model is too simplistic to capture the underlying patterns in the data.

From grade 12, the curves begin to diverge. The training MSE continues to decrease and eventually stabilises, while the test MSE begins to increase. This marks the transition to a low bias but high variance, an overfitting scenario in which the model becomes overly complex and begins to adapt to the noise in the training data, resulting in worsening generalisation to the unseen data.

Bias-Variance Trade-Off

We now examine the bias-variance trade-off in Ordinary Least Squares (OLS) regression by studying how the Mean Squared Error (MSE) varies with model complexity, represented here by the degree of the polynomial.

For each polynomial degree, we calculate three key metrics: error; bias; and variance. The results of this analysis, plotted for each polynomial degree, provide insights into how the balance between bias and variance shifts as model complexity increases.

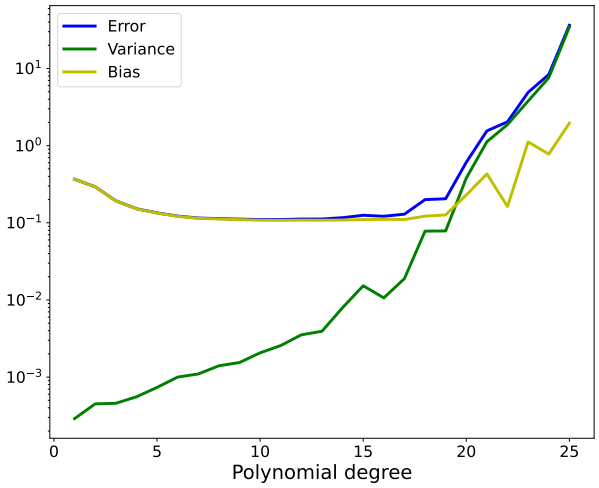

Figure 13: Error, bias and variance obtained applying bootstrap to OLS regression (1 000 observation)

3.1. Cross-validation

In order to conclude our analysis of the Franke function, we apply the cross-validation technique to evaluate once more the performance of our different models. In particular, we again calculate the MSE and then compare the results obtained for each model. We keep the noise weight at 0.1 and use the same 1 000 observations that characterized our analysis so far. Since for both Ridge and Lasso regression unscaled data lead to better results, we will apply cross-validation without scaling the data for any model. Finally, we use a number of folds equal to 5 for our evaluation. Figure **14** shows the results obtained.

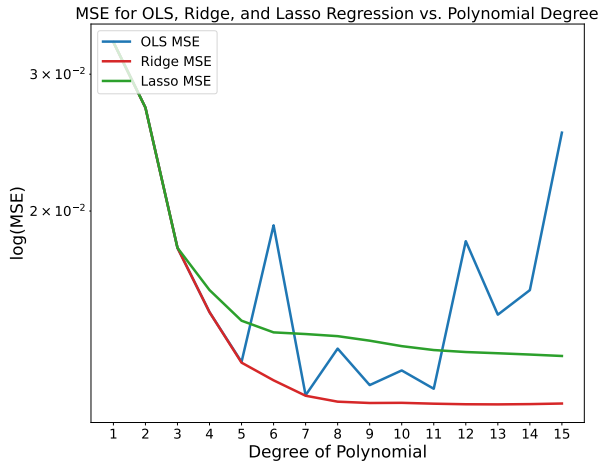

Figure 14: MSE results obtained by applying cross-validation to OLS, Ridge, and Lasso regression (1 000 observations).

It is immediately noticeable that the MSE for the OLS regression follows a rather erratic path. However, its characteristic U-shape is still visible: for larger degrees, the OLS regression tends to overfit. On the other hand, both the Ridge and Lasso regressions follow a more ordinary path. The regularization of the parameters can be appreciated by the more controlled behavior of the model for larger polynomial degrees. From this analysis, both the Ridge and Lasso regressions tend to perform better than the OLS one, with the Ridge regression having the best performance error-wise. This is an indication that L2 regularization may be more appropriate for this particular dataset.

Compared to the results obtained with the bootstrap analysis, the cross-validation technique shows a more unstable behavior of the OLS regression. However, both analysis indicate how the latter does not perform well for higher polynomial degrees, inevitably requiring some sort of regularization.

4. TERRAIN DATA

Ordinary Least Squares

After conducting preliminary tests using linear models on the Franke function, the analysis is extended to a real-world case study focusing on a specific region in Norway (Figure **2**). This dataset contains a large volume of geospatial data (6 485 401 data points) representing the morphology and topographic features of the terrain. Due to the substantial size of the original file, random sampling of smaller territorial segments is employed for preliminary experiments and model validation. This approach reduces computational costs while maintaining adequate data representativeness, ensuring that the results can be generalized to different contexts present in the complete dataset. Given a terrain dataset represented as a matrix $\mathbf{T}$ of size $n \times m$, where n and m correspond to the dimensions of the image:

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1m} \\ T_{21} & T_{22} & \cdots & T_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1} & T_{n2} & \cdots & T_{nm} \end{bmatrix}$$

we want to extract a random square sub-sample of size $s \times s$, where $s = 500$ (parameter that may vary depending on the analysis we are conducting). This requires choosing starting coordinates (x_0, y_0) such that:

$$0 \leq x_0 \leq m - s, \quad 0 \leq y_0 \leq n - s.$$

The extracted sub-sample $\mathbf{T}_{\text{sample}}$ is defined as:

$$\mathbf{T}_{\text{sample}} = \mathbf{T}[y_0 : y_0 + s,\, x_0 : x_0 + s].$$

The coordinates for the sampled region are represented using meshgrid arrays $\mathbf{X}_{\text{sample}}$ and $\mathbf{Y}_{\text{sample}}$, where:

$$\mathbf{X}_{\text{sample}} = \begin{bmatrix} 0 & 1 & \cdots & s-1 \\ 0 & 1 & \cdots & s-1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & s-1 \end{bmatrix}$$

$$\mathbf{Y}_{\text{sample}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ s-1 & s-1 & \cdots & s-1 \end{bmatrix}.$$

In Figure **15** we can see an example of this sampling technique. It's important to underling that the value of s is arbitrarily chosen.

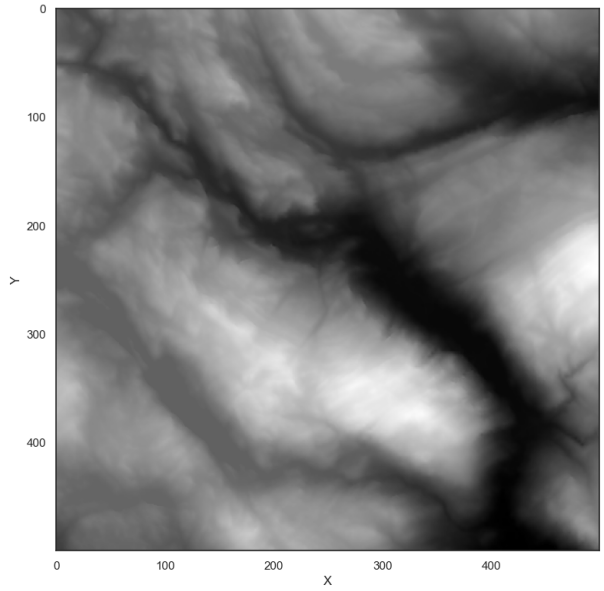

Figure 15: Grayscale matrix representing the original terrain data, with dimensions n x m, 500x500 pixel sample square region. `np.random.seed(7665324)`

The first thing that we do is, for each polynomial degree from 1 to the specified maximum, fit an OLS model. After fitting the models, we add a conditional statement to pick the best degree. The conditional statement checks whether the polynomial degree that maximizes R^2 is the same as the degree that minimizes MSE. If they are equal, that degree is considered the best; otherwise, the degree with the lowest MSE is selected. This prioritization of minimizing MSE over maximizing R^2 reflects a preference for a model that produces more accurate predictions, as MSE directly relates to the magnitude of prediction errors. This evaluation approach of tradeoff, ensures a balanced model that adequately explains the variance in the data while maintaining robust predictive performance. In OLS we find that the best degree that minimize the error and maximise the R^2 for that sample with `np.random.seed(234)`, between 1 and 30 degrees, is 12. In Figure **16** we have the graphical result.

Ridge

For the Ridge regression, we again follow the steps used for the Franke function. The sample size analyzed is the same as in the OLS analysis to allow a better comparability of the final results. We analyze different polynomial degrees (from 1 to 15) and different lambda values (40 different values ranging from 10^{-10} to 10^{10}). We loop for each polynomial degree, split the dataset at hand (we use the same random seed each time), and then use the `GridSearchCV` class from `scikit-learn` [1], allowing us to find the best lambda values for each polynomial degree. The results are shown in Table **3**.

D	Train MSE	Test MSE	Train R^2	Test R^2
1	30983.6507	30988.5000	0.0177	0.0174
2	12117.0812	12434.0934	0.6158	0.6057
3	5875.9521	5948.4044	0.8137	0.8114
4	5184.3835	5189.1728	0.8356	0.8355
5	4961.1093	4992.9170	0.8427	0.8417
6	3010.4800	3017.8835	0.9046	0.9043
7	2580.0064	2566.2919	0.9182	0.9186
8	2241.2507	2215.3022	0.9289	0.9298
9	2068.8706	2048.0869	0.9344	0.9351
10	1422.7417	1430.7000	0.9549	0.9546
11	1269.2526	1242.4819	0.9598	0.9606
12	10329.6440	10411.6970	0.6725	0.6698
13	6573.2533	6561.4753	0.7916	0.7919
14	3994.3227	4148.6746	0.8734	0.8684
15	18986.6553	18136.4356	0.3980	0.4249

Table 3: R^2 and MSE Values for Different Polynomial Degrees (40 000 observations) for Ridge regression on terrain data.

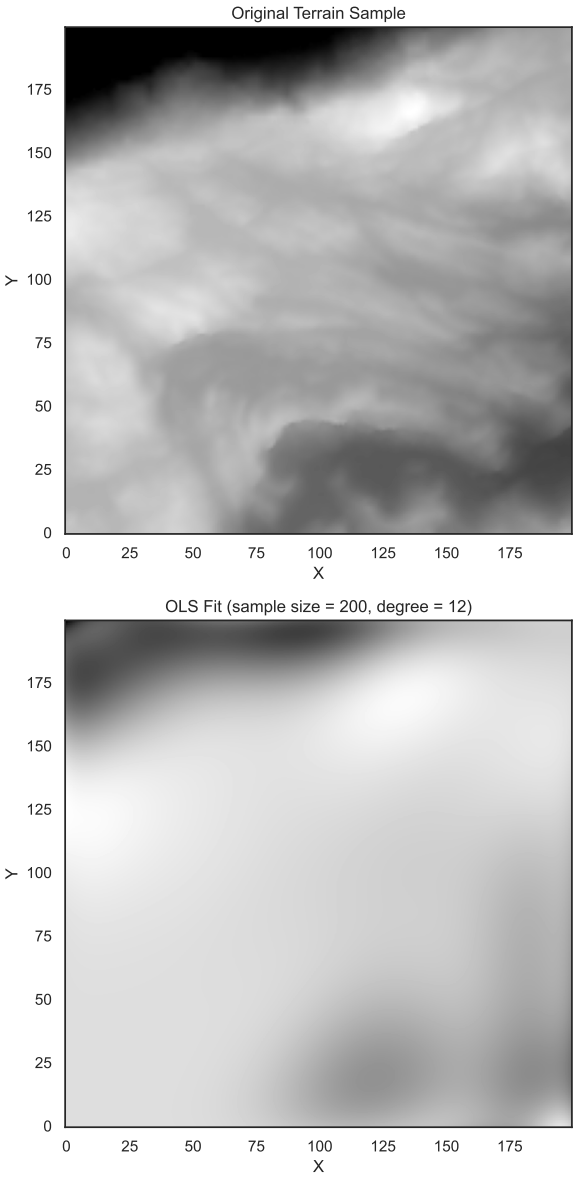

Figure 16: Grayscale original terrain data (250 000 pt) vs OLS fit (degree 12). Metrics: $R^2 = 0.93$, $MSE = 2258.77$. `np.random.seed(234)`

The first thing we can notice from the above results is that the MSE levels are significantly larger than what we have seen so far with the Franke function. On the other hand, since R^2 does not depend on the scale of the data, its values are more in line with the case analyzed before. This may indicate that our model still retains its ability to explain the variance in the data, but may be less accurate in its prediction given the more complex data to analyze. The model error starts to become too unstable aroud degree 12 (clearly shown by the MSE values for larger polynomial degrees in Table **3**). For this reason, our analysis does not evaluate polynomial degrees higher than 15, as this would not give us any more valuable insight. Finally, we can observe that the best overall model performance (both in terms of MSE and R^2) is achieved with a polynomial degree of 11.

Lasso

The analysis was extended to include Lasso Regression, following the same methodology applied to the previous models. Similar to the case of Ridge Regression, 20 values of λ were considered within the range $[10^{-5}, 10^{5}]$, selected using the logspace operator.

The evaluation procedure for the mean squared error (MSE) and the coefficient of determination (R^2) was consistent with that used for the Franke function. Given that a sample of 4 000 observations was retained, the process is particularly time-consuming, as the optimal value of α needs to be determined for each model degree. The results obtained are summarized in the figure **17**.

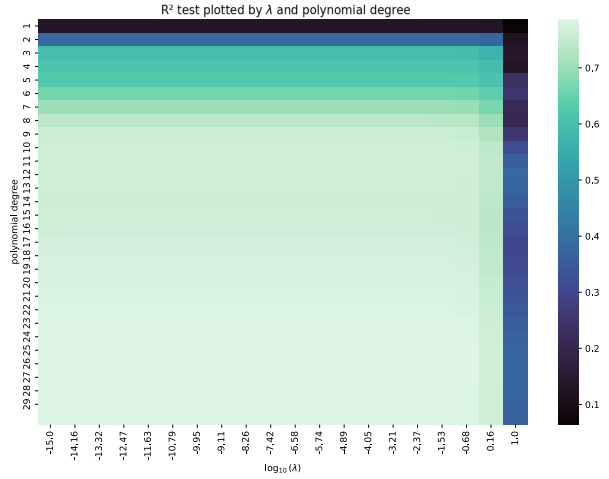

Figure 17: Test R^2 for different polynomial degrees (from 1 to 30) and lambda values (20 values ranging from 10^{-15} to 10^{1}), for a total of 4 000 observations for the terrain data.

The best result is obtained with a model degree of 30 and $\alpha = -0.26 = -1.411671 \times 10^{-6}$, resulting in a test MSE of 0.8964. As observed, the values obtained with Lasso are superior to those obtained with Ridge, indicating that Lasso performs better on this dataset.

Bias-Variance Tradeoff

The phenomenon of overfitting can be further analyzed by studying the decomposition of the MSE into Bias and Variance, as shown in figure **18**.

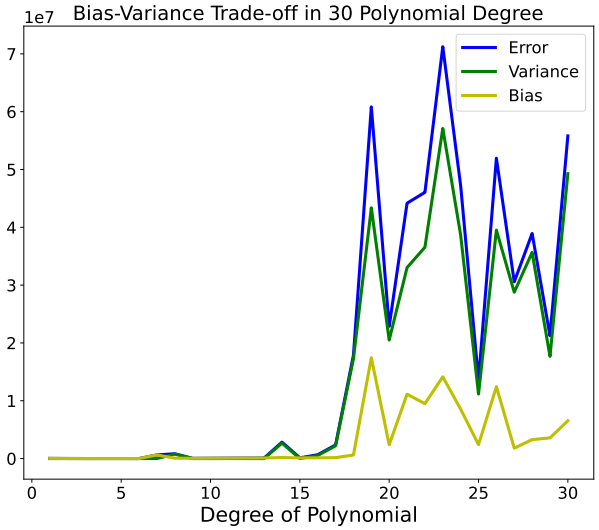

Figure 18: Error, bias and variance obtained applying bootstrap to OLS regression, on the Terrain data.

The previous figure illustrates the bias-variance trade-off. It can be observed that, for lower polynomial degrees, both variance and bias are close to zero. However, from degree 12 onwards, variance begins to increase and then exhibits oscillatory behavior for higher degrees. This pattern is once again attributed to the problem of overfitting.

While bias remains relatively stable, variance is significantly influenced by the model's tendency to overfit the training set. Consequently, a high variance implies that even small variations in the data lead to substantial changes in the model, particularly for high polynomial degrees. The turning point where overfitting becomes evident occurs around degree 12. This finding is consistent with the previous plots: the test MSE and R^2 start to deteriorate at approximately the same degree.

Cross-validaton

Finally, even for terrain data cross-validation represents the last step to follow in order to get a comprehensive overview of the different model behaviors. As previously done, we calculate the MSE for each model by applying a 5-fold cross-validation.

Given the complexity of the dataset at hand, we limit our analysis to a sample size of 1 000 datapoints and a range of polynomial degrees going from 1 to

10. We also examine a smaller set of lambda values, 20, ranging from 10^{-6} to 10^{6}. The results obtained can be seen in Table **4**.

Degree	OLS MSE	Ridge MSE	Lasso MSE
1	3775.8126	3775.8123	3775.8126
2	2810.6685	2810.6678	2810.6685
3	2534.8943	2534.8902	2534.8930
4	1905.1026	1905.0701	1924.6263
5	1569.2523	1569.2478	1707.7618
6	910.3658	910.3642	1540.8864
7	743.0764	743.1244	1265.2173
8	577.3148	602.1495	1067.1448
9	423.4075	530.9390	1019.5432
10	400.8394	451.9063	1014.6949

Table 4: MSE values obtained from CV for different polynomial degrees for OLS, Ridge, and Lasso regression (1 000 observations for terrain data).

A clear trend is recognizable even in this case: higher polynomial degrees correspond to better performances for all three models. In particular, the OLS regression shows the best final results, although they are very close to the Ridge results. On the other hand, the performance of the Lasso regression seems to be by far the worst of the three, indicating its unsuitability for the problem at hand.

5. CONCLUSIONS

In this study, we employed OLS, Ridge, and Lasso regression techniques to fit two-dimensional polynomial models. We initially tested these models on the Franke function before applying them to real terrain data from Norway. In particular, the number of datapoints can significantly affect the bias and variance of the model, as larger datasets typically lead to more stable and reliable parameter estimates. However, in this study, to ensure comparability across different regression models (OLS, Ridge, and Lasso), the size of the dataset has been kept constant at 1 000 observations.

OLS consistently achieved the lowest MSE when noise was minimal, and the dataset was sufficiently large. This indicates that for datasets where bias predominates and variance is limited, such as the terrain data, OLS proves to be the most reliable approach. Conversely, both Ridge and Lasso did not offer performance improvements in these conditions, as their regularization techniques are designed to mitigate variance, which was already low. However, we hypothesize that for higher-degree polynomial models, where the likelihood of overfitting increases due to elevated variance, Ridge and Lasso could outperform OLS by controlling variance more effectively.

When applied to the Franke function, the effectiveness of OLS, Ridge, and Lasso varied depending on the complexity of the polynomial model. For lower-degree polynomials, OLS provided superior results, whereas Ridge and Lasso became more effective as model complexity increased, where managing the bias-variance tradeoff became more essential. In high-variance scenarios, Lasso demonstrated some advantages, although its computational cost was higher due to the iterative nature of the algorithm.

In the case of terrain data, the low-noise environment favored OLS. However, Ridge exhibited some numerical instability, which warrants further investigation, while Lasso demonstrated better stability, likely due to its iterative approach, making it a potentially more robust alternative in certain circumstances.

OLS outperformed the other methods within the context of our dataset, but Ridge and Lasso remain valuable techniques in scenarios where variance management is critical, such as with noisier or more complex models. Future work could explore the application of these regularisation methods by varying the size of the data analysed, increasing the polynomial degree, or exploring alternative model structures, potentially leading to more accurate results in complex, real-world datasets.

Appendix A: Expectation values for OLS

We are going to show that the expectation value of y for a given element i is represented by:

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

and that its variance is:

$$\mathrm{Var}(y_i) = \sigma^2$$

Hence, $y_i \sim \mathcal{N}(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$, that is y follows a normal distribution with mean value $\mathbf{X}\boldsymbol{\beta}$ and variance σ^2.

We start by writing y_i in its matrix form:

$$y_i = \mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i$$

where

- $\mathbf{X}_{i,*}$ represents the i-th row of the matrix $\mathbf{X}$,
- $\boldsymbol{\beta}$ the parameters vector,
- $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

The expectation of y_i can be expressed using the linearity of expectation. Since $\mathbf{X}_{i,*}$ is non-stochastic, its expectation is simply itself, and $\mathbb{E}(\epsilon_i) = 0$.

$$\mathbb{E}(y_i) = \mathbb{E}(\mathbf{X}_{i,*}) + \mathbb{E}(\epsilon_i) = \mathbf{X}_{i,*}.$$

Next, we need to show that $\mathrm{Var}(y_i) = \sigma^2$. Since $\mathbf{X}_{i,*}$ is deterministic, its variance is 0. Thus, the variance of y_i is entirely due to the random variable ϵ_i, which has variance σ^2.

$$\mathrm{Var}(y_i) = \mathrm{Var}(\epsilon_i) = \sigma^2.$$

Appendix B: Expectation values for β parameters

With the OLS expressions for the optimal parameters $\hat{\boldsymbol{\beta}}$ we are going to show that:

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

and that the variance is:

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

We start by writing the expression for the ordinary least squares (OLS) estimator for $\hat{\boldsymbol{\beta}}$:

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

where $\mathbf{X}$ is the design matrix and $\mathbf{y}$ is the vector of observed values.

We use the linear model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

We substitute $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon}$ into the OLS estimator for $\hat{\boldsymbol{\beta}}$:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \\ &= \underbrace{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}}_{=\mathbf{I}}\boldsymbol{\beta} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon} \\ &= \boldsymbol{\beta} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}\end{aligned}$$

Then, we take the expectation of $\hat{\boldsymbol{\beta}}$:

$$\begin{aligned}\mathbb{E}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left(\boldsymbol{\beta} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}\right) \\ &= \boldsymbol{\beta} + \mathbb{E}\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}\right) \\ &= \boldsymbol{\beta} + \mathbb{E}((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\underbrace{\mathbb{E}(\boldsymbol{\epsilon})}_{=0} \\ &= \boldsymbol{\beta}\end{aligned}$$

In order to show that $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$:

$$\begin{aligned}\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathrm{Var}(\boldsymbol{\beta} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}) \\ &= \mathrm{Var}\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}\right) \quad (\mathrm{Var}(\mathbf{A}\boldsymbol{\epsilon}) = \mathbf{A}\,\mathrm{Var}(\boldsymbol{\epsilon})\mathbf{A}^T) \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\underbrace{\mathrm{Var}(\boldsymbol{\epsilon})}_{=\sigma^2\mathbf{I}}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} \\ &= \sigma^2\underbrace{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}}_{=\mathbf{I}}(\mathbf{X}^T\mathbf{X})^{-1} \\ &= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\end{aligned}$$

Appendix C: Bias-variance trade off derivation

Consider a dataset $\mathcal{L}$ consisting of the data $\mathbf{X}_\mathcal{L} = \{(y_j, \mathbf{x}_j), j = 0 \dots n-1\}$.

We assume that the true data is generated from a noisy model

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

Here ϵ is normally distributed with mean zero and standard deviation σ^2.

In our derivation of the ordinary least squares method, we defined an approximation to the function f in terms of the parameters $\boldsymbol{\beta}$ and the design matrix $\mathbf{X}$ which embody our model, that is $\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$.

The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the mean squared error via the so-called cost function

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

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

We are going to show that:

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

with the bias component equal to:

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

and the variance:

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

First, it is worth noting that by rewriting the cost function as an expectation (rather than a sample mean), we are generalizing it. That is, we are considering a broader probabilistic framework, not just a finite dataset, but a distribution over all possible outcomes (the true distribution of $\mathbf{y}$). So we obtain the following equation:

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

This notation also allows us to break the total error into bias, variance, and noise components for the true model (which we can't directly observe) rather than just the error on a single dataset.

In order to show that:

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

we start by writing the cost function as an expactation:

$$C(\mathbf{X}, \boldsymbol{\beta}) = \mathbb{E}\left[(\mathbf{y} - \tilde{\mathbf{y}})^2\right]$$

We expand $(\mathbf{y} - \tilde{\mathbf{y}})^2$ using the fact that $\mathbf{y} = f(\mathbf{x}) + \epsilon$:

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

Then, we add and subtract $\mathbb{E}[\tilde{\mathbf{y}}]$ to the above equation:

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

Finally, by applying the linearity of expectation, we can rewrite the above formula as follows:

$$\begin{aligned}\mathbb{E}\left[(\mathbf{y} - \tilde{\mathbf{y}})^2\right] = \mathbb{E}\left[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2\right] \\ + \mathbb{E}\left[(\tilde{\mathbf{y}} - \mathbb{E}[\tilde{\mathbf{y}}])^2\right] + \sigma^2.\end{aligned} \tag{20}$$

Now we can rewrite the terms as:

- $\text{Bias}[\tilde{\mathbf{y}}] = \mathbb{E}\left[(f(\mathbf{x}) - \mathbb{E}[\tilde{\mathbf{y}}])^2\right]$. It measures how much the average prediction ($\mathbb{E}[\tilde{\mathbf{y}}]$) differs from the true function value ($f(\mathbf{x})$).
- $\text{Var}[\tilde{\mathbf{y}}] = \mathbb{E}\left[(\tilde{\mathbf{y}} - \mathbb{E}[\tilde{\mathbf{y}}])^2\right]$. It measures how much the predictions $\tilde{\mathbf{y}}$ vary around their expected value $\mathbb{E}[\tilde{\mathbf{y}}]$.
- Noise term: σ^2. This is the irreducible noise in the data, that we assume follows a normal distribution.

Thus, the final expression is:

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

References

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

[2] S. Raschka et al. *Machine Learning with PyTorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*. Expert insight. Packt Publishing, 2022. ISBN: 9781801819312. URL: https://books.google.no/books?id=UHbNzgEACAAJ.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[4] *Norgeskart*. norgeskart.no. URL: https://norgeskart.no/#.

[5] P. A. Lopez Torres G. Durante E. Ottoboni. *Project1*. https://github.com/Elisaottoboni/Machine-Learning-University-of-Oslo/tree/main/Project_1. 2024.

[6] M. Hjorth-Jensen. *MachineLearning*. https://github.com/CompPhysics/MachineLearning. 2024.