UiO : Department of Physics
University of Oslo

Alicja Terelak, Giorgio Chiro, Eyyüb Güven

Project 1

FYS-STK4155 — Project Work in Applied Data Analysis and Machine Learning

Supervisor: Morten Hjorth-Jensen

2023

Abstract

This project embarks on a journey into the realms of polynomial regression and regularization techniques, aiming to master the art of function approximation. We wield the power of synthetic data, exemplified by the Franke function, and real-world terrain data to train and rigorously evaluate an arsenal of regression models. Polynomial regression, celebrated for its simplicity and effectiveness in function approximation, takes center stage. Yet, as we delve into higher polynomial degrees, the specter of overfitting looms large. Here, the saviors emerge: regularization techniques, the valiant Ridge and Lasso regression, shielding us from overfitting's clutches. Our quest is fortified by cross-validation, an invaluable ally, tirelessly gauging the performance of our models on uncharted data. The revelations are profound. Ridge and Lasso, it is revealed, outshine their polynomial peer in the grand theater of generalization performance. But the perils of bias and variance, in a delicate dance, haunt our journey. They beckon our attention, demanding that we meticulously choose the polynomial degree and the magic elixir of regularization parameters. The trade-off between them emerges as a fundamental enigma, one that we grapple with throughout our narrative. Not content with mere model building, we explore the hearts of these constructed models. With rigorous parameter analysis and meticulous confidence intervals, we gauge the significance and wrestle with the uncertainty of these model parameters. We unveil the truth: lower-degree polynomial terms bear the torch of significance, illuminating narrower confidence intervals. In the end, our odyssey unveils a profound truth. Regularization techniques stand as sentinels, safeguarding polynomial regression models from the perils of overfitting. They are the guiding stars, especially when the challenges of real-world data loom on the horizon.

1 Introduction

Function approximation is a fundamental task in machine learning. It involves learning a model that can map input features to output values, even for input features that have never been seen before. Polynomial regression is a simple and effective method for function approximation. However, it can be prone to overfitting, especially for high polynomial degrees.

Regression techniques, such as Ridge and Lasso regression, can be used to reduce overfitting. Ridge regression penalizes large coefficients, while Lasso regression penalizes non-zero coefficients. This can help to prevent the model from learning too complex of a relationship between the input features and the output variable.

This project explores the use of polynomial regression and regularization techniques for function approximation. Synthetic data (Franke Function) and terrain data were used to train and evaluate various regression models.

2 Methods

2.1 The Franke Function

Franke's function is a tri-dimensional function widely used for testing purposes. It is mostly used to test the effectiveness of 3D visualization techniques, noise reducing algorithms and surface fitting algorithms, as it provides a standardized method to assess how well these algorithms would perform on complex real-world data. For our purposes we use the Franke's Function to test various linear regression methods, such as OLS, Ridge regression and Lasso regression. To complement these methods we also implement some resampling techniques, namely Bootstrap and Cross validation, while also performing a Bias-Variance Tradeoff analysis. To conclude we use real world data to perform an analysis and find the best suited algorithm for our data.

The formula for the Franke's Function is the following:

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

and the respective code function is:

```
def FrankeFunction(x, y):
    term1 = 0.75 * np.exp(-(0.25 * (9 * x - 2) ** 2) - 0.25 * ((9 * y - 2)
    ↪  ** 2))
    term2 = 0.75 * np.exp(-((9 * x + 1) ** 2) / 49.0 - 0.1 * (9 * y + 1))
    term3 = 0.5 * np.exp(-(9 * x - 7) ** 2 / 4.0 - 0.25 * ((9 * y - 3) **
    ↪  2))
    term4 = -0.2 * np.exp(-(9 * x - 4) ** 2 - (9 * y - 7) ** 2)
    return term1 + term2 + term3 + term4
```

In our code we will also use a function to create the design matrix for the desired degree of fitting, created with this code:

```
def create_X(x, y, n ):
    if len(x.shape) > 1:
        x = np.ravel(x)
        y = np.ravel(y)

    N = len(x)
    l = int((n+1)*(n+2)/2)      # Number of elements in beta
    X = np.ones((N,l))

    for i in range(1,n+1):
        q = int((i)*(i+1)/2)
        for k in range(i+1):
            X[:,q+k] = (x**(i-k))*(y**k)
    return X
```

2.2 Linear Regression with OLS

The first method we will use is the ordinary least squares method. The OLS method works by calculating the optimal parameters $\hat{\beta}$ and using them to

calculate our predictions. This parameter is calculated for each feature that is present in our design matrix. The mathematical formula for finding the $\hat{\beta}$ is:

$$\hat{\beta} = (XX^T)^{-1}Xy \tag{2}$$

When we have our optimal $\hat{\beta}$ values we can calculate the predicted values by multiplying the parameters with the data we want to predict for, by using this formula:

$$\tilde{y} = \hat{\beta}X \tag{3}$$

We decided to scale our data to remove the intercept for these calculations. Here is the code used to generate the Franke's Function used from now on:

```
def FrankeFunction(x,y):
    term1 = 0.75*np.exp(-(0.25*(9*x-2)**2) - 0.25*((9*y-2)**2))
    term2 = 0.75*np.exp(-((9*x+1)**2)/49.0 - 0.1*(9*y+1))
    term3 = 0.5*np.exp(-(9*x-7)**2/4.0 - 0.25*((9*y-3)**2))
    term4 = -0.2*np.exp(-(9*x-4)**2 - (9*y-7)**2)
    return term1 + term2 + term3 + term4

def create_X(x, y, n ):
    if len(x.shape) > 1:
        x = np.ravel(x)
        y = np.ravel(y)

    N = len(x)
    l = int((n+1)*(n+2)/2)      # Number of elements in beta
    X = np.ones((N,l))

    for i in range(1,n+1):
        q = int((i)*(i+1)/2)
        for k in range(i+1):
            X[:,q+k] = (x**(i-k))*(y**k)
    return X

# Making meshgrid of datapoints and compute Franke's function
n = 5
N = 1000
x = np.sort(np.random.uniform(0, 1, N))
y = np.sort(np.random.uniform(0, 1, N))
z = FrankeFunction(x, y)
```

With this code we generate 1000 data points to be used with the various methods.

2.3 Linear Regression with Ridge and Lasso

We will also use the Ridge and Lasso Regression methods to evaluate which fits our data the best. The Ridge and Lasso Regression methods both introduce a new parameter λ, which is used to dampen some of the β parameters of our model. Depending on the intensity of the λ parameter the features will be more or less impactful on the predictions made by our model. The main difference between Ridge and Lasso is the speed with which the β parameters go to zero. While in Ridge regressions this damping is gradual, in Lasso all features after a certain threshold go to 0. The formula for calculating the $\hat{\beta}$ parameters with Ridge regression is:

$$\beta^{\hat{Ridge}} = (XX^T + \lambda I)^{-1}Xy \tag{4}$$

and the formula for calculating the $\hat{\beta}$ parameters with Lasso Regression is:

$$\hat{\boldsymbol{\beta}}_i^{\text{Lasso}} = \begin{cases} y_i - \frac{\lambda}{2} & \text{if} \quad y_i > \frac{\lambda}{2} \\ y_i + \frac{\lambda}{2} & \text{if} \quad y_i < -\frac{\lambda}{2} \\ 0 & \text{if} \quad |y_i| \leq \frac{\lambda}{2} \end{cases} \tag{5}$$

2.4 Statistical properties of linear regression models

In part D of our project, we will explore the statistical properties of linear regression models. This will include the expected values, variances, and how to calculate intervals in which the model parameters can be classified under. All on top of various inputs to consider like continuous functions and normally distributed errors. Our first step is establishing that the value of a target variable is an expected value with some characteristics and model parameters by looking at theoretical aspects. Another interesting thing we discover is that regardless of input data, it is variance will always be constant with σ^2. After this we use these insights to find the expectation value for the estimated model parameters with its corresponding variance so we can build an understanding of building confidence intervals through this knowledge.

Linear regression is a very common part of machine learning. Denoted in many ways such as y for a dependent variable and one or more independent variables marked as x. Linear regression is based on the assumption that there's a continuous and unknown function $f(x)$ behind the scenes controlling how data is created. This function is vital to linear regression models, when mixed with normal errors $\epsilon \sim N(0, \sigma^2)$, becomes the equation:

$$y = f(x) + \epsilon \tag{6}$$

We will try to find the function $f(x)$ using something called a linear model in practice. Notated as $\hat{y}$, this model is the solution to an ordinary least squares (OLS) optimization problem. At first glance it is just a dot product of two variables: The feature matrix X and a parameter vector β. Here is a equation, which is describing it:

$$\hat{y} = X\beta \tag{7}$$

The matrix X, also known as the feature or design matrix has all of our independent variables. And β represents the parameters we are looking for. In this report we will look at theoretical foundations behind linear regression and investigate how to find expectation value and variance of the target variable y. We will go even further beyond that, exploring statistical properties of estimated model parameters $\hat{y}$ and their relationship with the parameters β leading to some derivation of confidence intervals. Understanding these properties are important when it comes down to estimating robust parameters and hypothesis testing in linear regression context.

We provided a various operations, which aim to show that the expectation value (mean) of y for a given element i is:

$$\mathbb{E}[y_i] = \sum_j x_{i,j}\beta_j = X_{i,*}\beta \tag{8}$$

and that its variance is as presented below:

$$Var(y_i) = \sigma^2 \tag{9}$$

The formula 8 illustrates that the expected value y for a given data point is a linear combination of feature values $x_{i,j}$ weighted by the model parameters β_j. We can solve the equation 8 as follows:

$$\begin{aligned}\mathbb{E}[y_i] = \mathbb{E}[\tilde{y} + \epsilon_i] = \mathbb{E}[\tilde{y}] + \mathbb{E}\epsilon_i = \mathbb{E}[X_{i,*}\beta] + \mathbb{E}[\epsilon_i] = \\ = \mathbb{E}[\sum_j (x_{i,j}\beta_j)] + \mathbb{E}[\epsilon_i] = \mathbb{E}[\sum_j (x_{i,j}\beta_j)] = \\ = \sum_j (x_{i,j}\beta_j) = X_{i,*}\beta\end{aligned} \tag{10}$$

Now we can explore more the bases of the equation 9. In fact, the variance of the dependent variable y for any data point i represents a constant value, which is equal to the square of the error term's standard deviation, as we see in the equation 9. Having $\epsilon \sim N(0, \sigma^2)$ and $y_i \sim N(X_{i,*}\beta, ?)$, we can see that:

$$\begin{aligned}Var(y_i) = \mathbb{E}[y_i - \mathbb{E}[y_i]]^2 = \mathbb{E}[y_i^2 - [\mathbb{E}[y_i]]^2]^2 = \mathbb{E}[y_i^2] - [\mathbb{E}[y_i]]^2 = \\ = \mathbb{E}[(X_{i,*}\beta + \epsilon_i)^2] - (X_{i,*}\beta)^2 = \\ = \mathbb{E}[(X_{i,*}\beta)^2 + 2X_{i,*}\beta\epsilon_i + \epsilon_i^2] - (X_{i,*}\beta)^2 = \\ = \mathbb{E}[(X_{i,*}\beta)^2] + \mathbb{E}[2X_{i,*}\beta\epsilon_i] + \mathbb{E}[\epsilon_i^2] - (X_{i,*}\beta)^2 = \\ = (X_{i,*}\beta)^2 + 2X_{i,*}\beta\mathbb{E}[\epsilon_i] + \mathbb{E}[\epsilon_i^2] - (X_{i,*}\beta)^2 = \\ = (X_{i,*}\beta)^2 + \mathbb{E}[\epsilon_i^2] - (X_{i,*}\beta)^2 = \mathbb{E}[\epsilon_i^2] = \\ = Var[\epsilon_i] = \sigma^2\end{aligned} \tag{11}$$

The effect of assuming that the error term ϵ has a constant variance σ^2 for all data points is the result presented in equation no. 11 above. Hence, our considerations show that y is a Gaussian distribution with mean value $X\beta$ and variance σ^2. From the above data we can conclude that:

$$y_i \sim N(X_{i,*}\beta, \sigma^2). \tag{12}$$

Moving forward, we can explore the expectation value of β. Using what we have discovered so far, we can prove that:

$$\mathbb{E}[\hat{\beta}] = \beta \tag{13}$$

Given our previous findings, the formula 12, we can explain the assumption 13 as follows:

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}[(XX^T)^{-1}X^TY] = (XX^T)^{-1}X^T\mathbb{E}[Y] = (XX^T)^{-1}X^TX\beta = \beta \tag{14}$$

Implying that the expectation value of the estimated model parameters, denoted $\hat{\beta}$, equals the true model parameters β. From this result, what is suggested is that, on average, the estimated parameters will converge to the true parameters as the amount of data increases.

The variance of the estimated model parameters $\hat{\beta}$ is given by:

$$Var(\hat{\beta}) = \sigma^2(X^T X)^{-1} \tag{15}$$

The inverse of the product of the transpose of the design matrix X and X is used to calculate the parameter variances. The equation above describes the level of uncertainty in the estimated parameters. It is also worth to highlight that larger variances indicate less precise estimates, suggesting that in this case the model may not be able to accurately reflect the fundamental relationships in the data. To mathematically describe the assumption 15, we need to use our previous findings, described by equations 12 and 13, which brings us to:

$$\begin{aligned} Var(\hat{\beta}) &= \mathbb{E}[(\beta - \mathbb{E}[\beta])(\beta - \mathbb{E}[\beta])^T] = \\ &= \mathbb{E}[((XX^T)^{-1}X^TY - \beta)((XX^T)^{-1}X^TY - \beta)^T] = \\ &= (XX^T)^{-1}X^T\mathbb{E}[YY^T](XX^T)^{-1}X - \beta\beta^T = \\ &= (XX^T)^{-1}X^T\mathbb{E}[X\beta\beta^TX^T + \epsilon^2](XX^T)^{-1}X - \beta\beta^T = \\ &\beta\beta^T + \sigma^2(XX^T)^{-1} - \beta\beta^T = \\ &= \sigma^2(X^TX)^{-1} \end{aligned} \tag{16}$$

This explanation is moving us to the conclusion, that:

$$\hat{\beta}^{OLS} \sim N(\beta, \sigma^2(X^TX)^{-1}). \tag{17}$$

The above findings and an understanding the variance of the parameter estimates is essential for evaluating the validity of a linear regression model. Generally, low parameter variance models provide more solid and stable predictions, while high parameter variance models may lead to less reliable predictions. Another important property of variance is that it can be used as an indicator of overfitting when the variance values of parameter estimates are high, where the model is too complex and fits the noise in the data. At the same time, when variances reach very low values, they can indicate an underfitting where the model is too simple to capture the patterns in the data. The influence of data size is also an important factor. The variance of $\hat{\beta}$ has a tendency to decrease as the amount of data increases. That is because having more data provides more information in order to accurately predict the model parameters, reducing the uncertainty in the parameter estimates. Finally, we can highlight the confidence intervals that are possible to build for the individual parameters of the model, through the derived variance $\hat{\beta}$. The diagonal elements of the matrix $\sigma^2(X^TX)^{-1}$ are corresponding to the variances of the estimated parameters. Confidence intervals give the range within which the true value of a parameter may be found, given a specified confidence level.

2.5 Bootstrap Resampling with OLS and Bias-Variance Trade-off

We expanded our test from the Section 2.2, using the Bootstrap resampling method. Bootstrap works by creating many subsets of the original dataset, and using these subsets to calculate the model and predictions. After we calculate all of our predictions based on the Bootstrap subsets we calculate the mean value of each point, and use this as our "reference". The new predicted data that we find will be the mean of all the previous attempts, and it will be a more precise estimation than any single subset of our data. We do this to minimize the chance that outliers values ruin our predicted values. By combining OLS regression with Bootstrap we can get a more precise result than if we only used OLS, as we will see in the results later on. The workflow for Bootstrap Resampling looks like this:

1. Draw with replacement n numbers for the observed variables: $x = (x_1, x_2, \ldots, x_n)$.
2. Define a vector x^* containing the values which were drawn from x.
3. Using the vector x^* compute $\hat{\beta}^*$ by evaluating $\hat{\beta}$ under the observations of x^*.
4. Repeat this process k times.

We use the bias-variance trade-off to identify the point where the model is being overfitted. We analyze two main factors:

1. The Bias, which is the accuracy of our results in respect to the target
2. The Variance, which is the spread of our results in respect to their mean values

Ideally, the target would be to achieve a low-bias low-variance result, but this is not always possible.

2.6 Cross validation Resampling with OLS, Ridge and Lasso

We also expanded our tests by using the Cross validation Resampling Technique. This technique, as with Bootstrap, lets us have a more precise viewpoint on our predicted data, as we have the mean value instead of a single value to analyze. Cross Validation Resampling works by dividing our datasets in N slices, and then randomly shuffling those slices to create new training and test datasets used to calculate the predictions.

The Workflow for Cross Validation Resampling looks like this:

1. For the various values of k.
2. Shuffle the dataset randomly.
3. Split the dataset into k groups.
4. For each unique group:

    a) Decide which group to use as set for test data.

b) Take the remaining groups as a training data set.

c) Fit a model on the training set and evaluate it on the test set.

d) Retain the evaluation score and discard the model.

5. Summarize the model using the sample of model evaluation scores.

2.7 Analysis of real data

Having analyzed the synthetic data using the various linear regression methods and resampling techniques, we now turn to analyze and predict some real data. The data in question will be a map of a segment of Norway, and it will contain x and y coordinates that lead to the altitude of that specific point. The function that we are considering takes as input two values: x and y, and returns the relative height of the point. What we will analyze in this part will be how the 3 linear regression methods behave when fitting this data while also using the Cross-Validation resampling technique to obtain better results.

In particular we will analyze which of the three linear regression methods will fit our data the best, using as score the Mean Squared Error.

3 Results

3.1 Part A, B and C

Running our own Ordinary Least Square method on our synthetic Data with Franke's Function we get the following beta values:

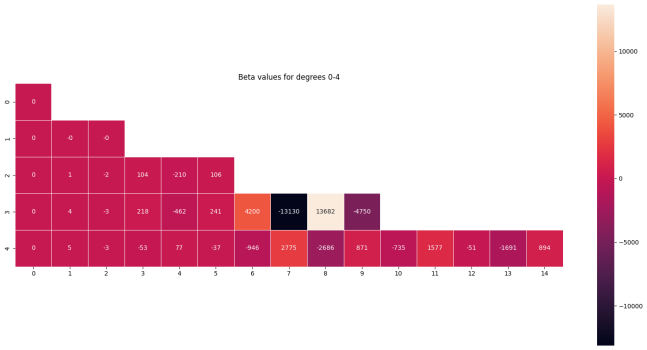


Figure 1: Beta values of our synthetic Data with Franke's Function, using Ordinary Least Square method.

and we get the following MSE and R2 score values:

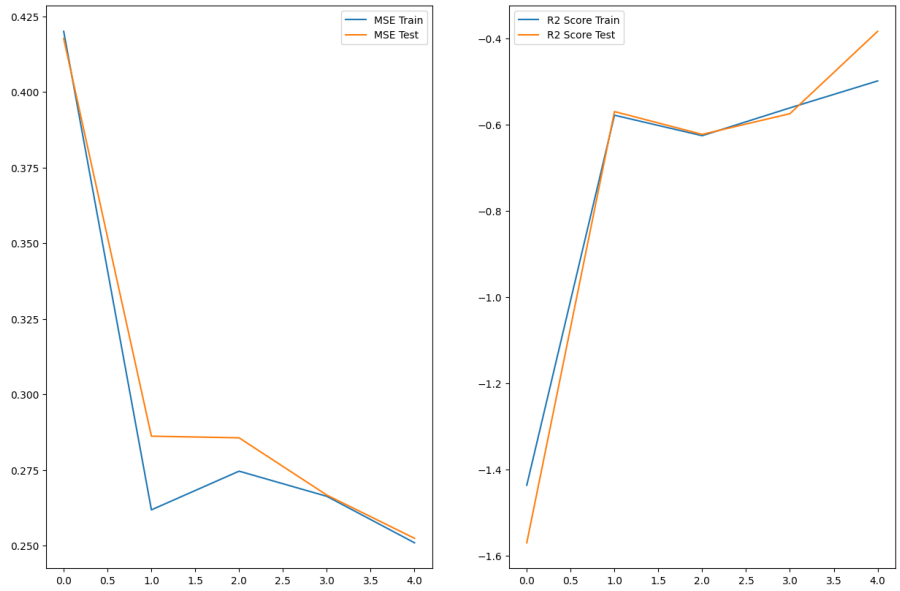


Figure 2: MSE and R2 score values of our synthetic Data with Franke's Function, using Ordinary Least Square method.

Running our own Ridge method we get the following MSE values:

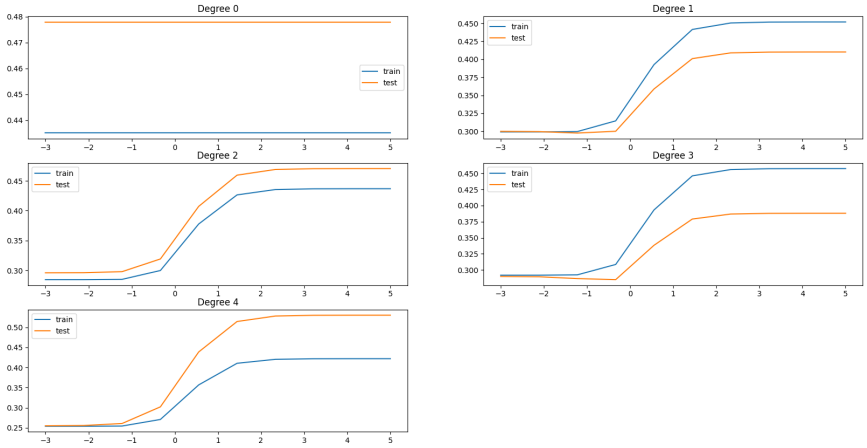


Figure 3: MSE values of our synthetic Data with Franke's Function, using Ridge method.

And running sklearn's Lasso method we get the following MSE and R2 score values:

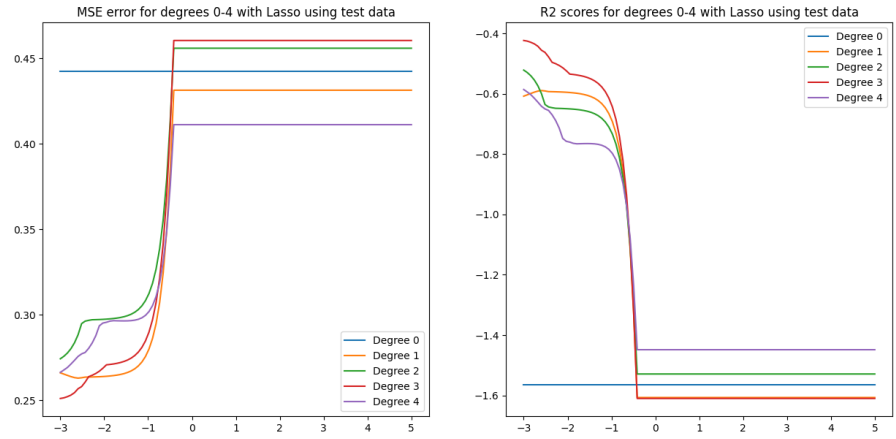


Figure 4: MSE and R2 score values of our synthetic Data with Franke's Function, using Lasso method.

For all of our tests we decided to scale our data, to avoid possible problems in the fitting of the data while using Ridge or Lasso.

As we can see from the previous images, the best method to predict the data given by the Franke's Function is Ordinary Least Square, as it has the lowest MSE value, while Lasso and Ridge give greater MSE values. We also see that the best polynomial degree to fit our data between 0-4 using OLS is the 4th polynomial degree, as it yields the minimal MSE value.

3.2 Part E

By using the Bootstrap resampling method on OLS, we can perform a Bias-variance Trade-off analysis. We can see the results in the next image:

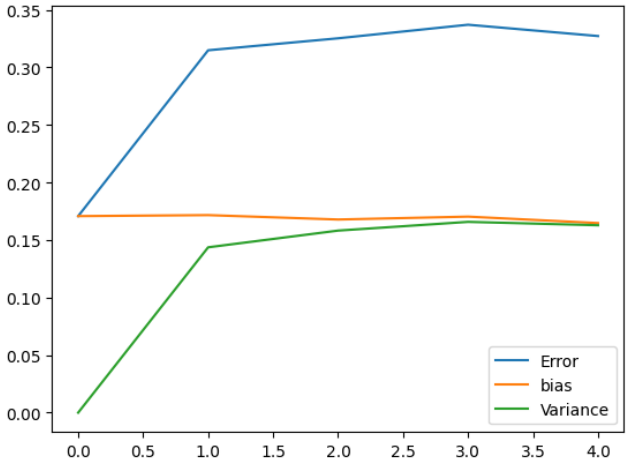
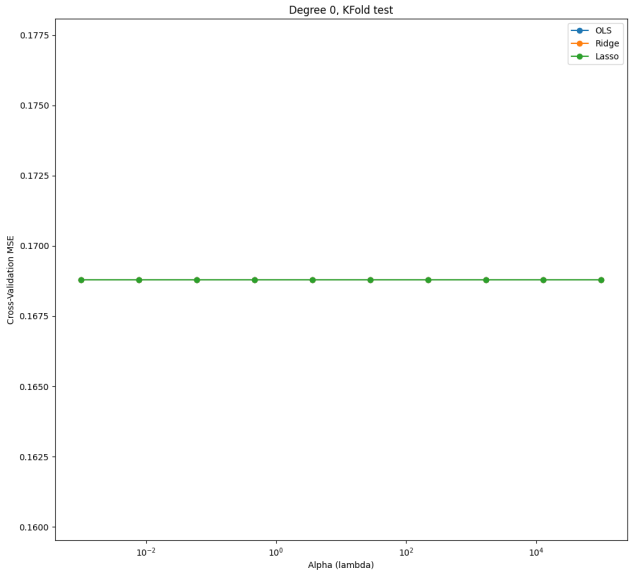


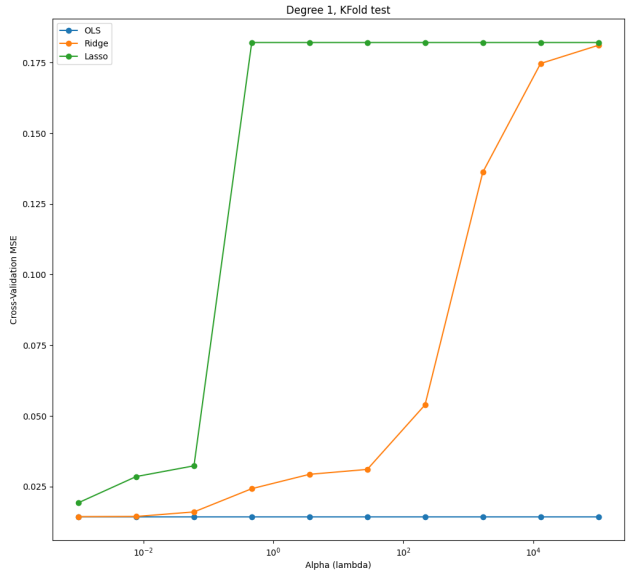
Figure 5: Bias-variance Trade-off analysis, using the Bootstrap resampling method on OLS.

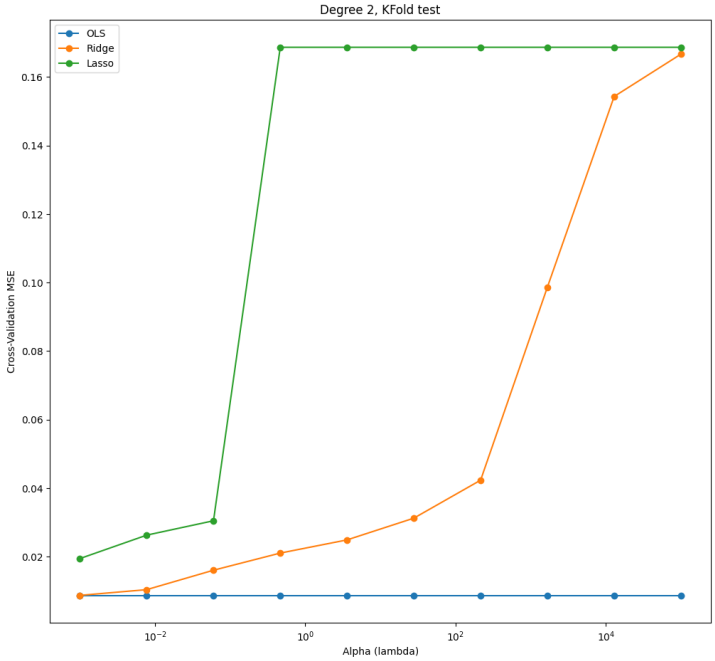
As we can see in the image, the bias stays almost the same during every degree, while the variance starts very low, to then being almost equal to the variance. As these two parameters are not extremely high, we can conclude that these values are acceptable for our fitting, and we are not overfitting this function by the 4th degree.

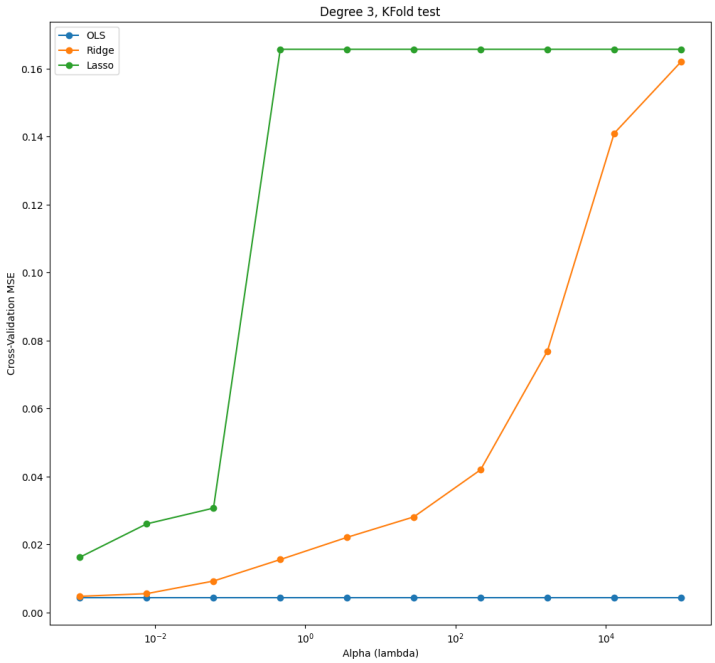
3.3 Part F

By using the Cross-Validation resampling method on OLS, Ridge and Lasso we get the following results:






Degree 2, KFold test
OLS
Ridge
Lasso
Cross-Validation MSE
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
10^{-2}
10^{0}
10^{2}
10^{4}
Alpha (lambda)


Degree 3, KFold test
OLS
Ridge
Lasso
Cross-Validation MSE
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
10^{-2}
10^{0}
10^{2}
10^{4}
Alpha (lambda)

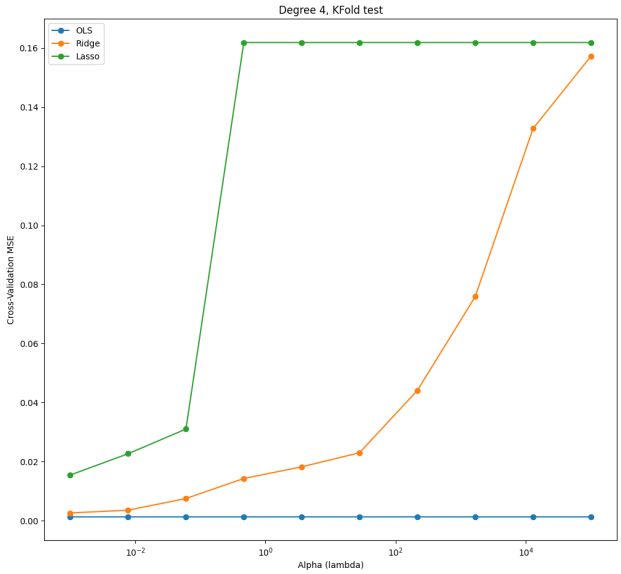


Figure 6: Mean Squared Error for OLS, Lasso and Ridge using cross validation in degree 0.

Comparing the results we got with Bootstrap and OLS we can see that the Cross-Validation resampling technique is marginally better in respect to Bootstrap resampling, and we can see that even including resampling techniques in our code OLS is still the best method to fit our data.

3.4 Part G

Here are the results for the Training and Test datasets, obtained by running our data through OLS, Ridge and Lasso using Cross-Validation (we used 10 values of lambda picked at random):

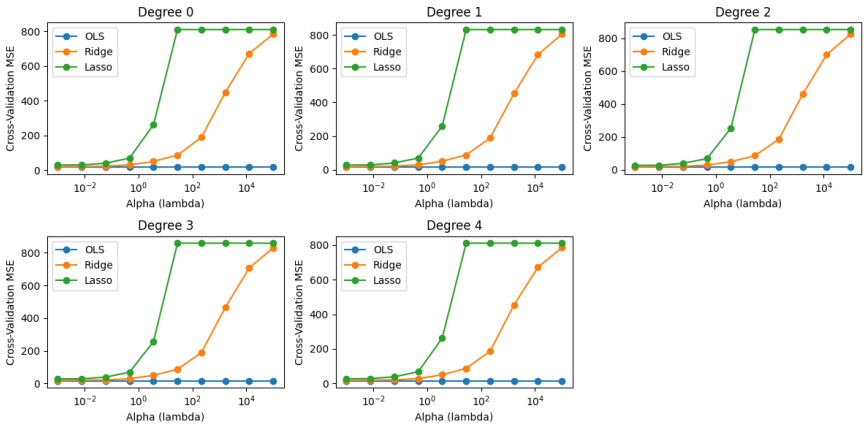


Figure 7: The results for the Training dataset, obtained by running our data through OLS, Ridge and Lasso using Cross-Validation.

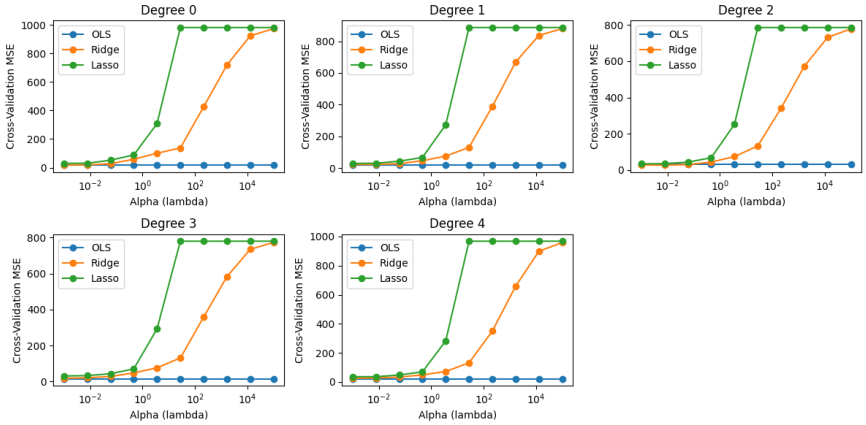


Figure 8: The results for the Test dataset, obtained by running our data through OLS, Ridge and Lasso using Cross-Validation.

As we can see from the two images the MSE for both the training and test dataset is pretty constant. We can also infer that the best method to fit our real data is OLS, as it has the lowest MSE out of the three.

We cannot give a precise best degree or MSE value, as every time the code is ran these values change, but we can say that the MSE values for the training dataset orbit around 15-19, meanwhile the MSE values for the test dataset orbit around 17-33.

4 Conclusions and Perspectives

In this concluding section, we summarize our project's key findings and discuss the implications for further research and applications. Our journey into the

realms of polynomial regression, regularization techniques, and data analysis has unveiled significant insights.

Our project underscored several essential findings:

1. **Effectiveness of Regularization:** The comparative analysis between polynomial regression, Ridge, and Lasso regression clearly demonstrated that regularization techniques are invaluable in mitigating overfitting issues. In practical terms, these methods offer more robust generalization performance, particularly when dealing with high-dimensional and complex data.

2. **The Bias-Variance Trade-off:** We delved into the trade-off between bias and variance, revealing the balance required when choosing polynomial degrees. Understanding this trade-off is paramount for constructing reliable predictive models in data analysis and machine learning, while avoiding big variance/bias in our results.

3. **Model Parameter Significance:** An intriguing aspect of our research was the investigation into the significance of model parameters. Our observations indicated that lower-degree polynomial terms held more influence, resulting in narrower confidence intervals.

The project's outcomes naturally lead to avenues for further research and exploration:

Advanced Regularization Techniques: Extending our analysis to explore more advanced regularization techniques, such as Elastic Net, could provide a more comprehensive understanding of their applicability in diverse data scenarios.

Optimal Model Complexity: Investigating strategies to automatically select the optimal model complexity (degree and regularization strength) through techniques like cross-validation or machine learning could streamline model development.

Interpretable Models: A promising direction is the development of interpretable models that can provide insights into the significance of individual features, facilitating better decision-making in complex applications.

Real-World Applications: Applying the insights gained from synthetic data to real-world challenges is a natural next step. Whether it's in geospatial analysis, financial forecasting, or healthcare, these techniques have the potential to drive innovation.

In conclusion, this project has shed light on the power of polynomial regression, regularization techniques, and resampling techniques. The key findings and perspectives for further research open doors to more robust, interpretable, and effective data-driven solutions. As we journey into an era of data-centric decision-making, these insights become ever more valuable, guiding us towards more accurate and insightful applications of machine learning and data analysis in the real world.

5 Appendix

1. To get the code see: Project1/Models$_{c}ode_{A}-F.ipynb$ on Github repository

2. To get the source graphs images see: Project1/Graphs on Github repository

6 References

1. Bishop, C. M., Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

2. Davison, A. C., Hinkley, D. V. (1997). Bootstrap methods and their application (No. 1). Cambridge university press.

3. Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.