Applied Data Analysis and Machine Learning

Regression Analysis and Resampling Methods

Mohamad Mahmoud & Aleksandar Davidov
GitHub - click here -

Oct 2021

Abstract

A regression analysis on both pseudo data using the Franke's function and terrain data from earthexplorer.usgs.gov. The regression models in question are Ordinary Least Squares(OLS), Ridge and Least Absolute Shrinkage and Selection Operator (LASSO), respectively. For both OLS and Ridge we have developed and tested our own algorithms using `Numpy` and compared them to their corresponding counterparts from `Scikit`. LASSO was implemented by using the available LASSO method in `Scikit`.

Contents

I. INTRODUCTION

The goal of this project is to implement and compare the three linear regression methods, Ordinary Least Squares (OLS), Ridge and Least Absolute Shrinkage and Selection Operator (LASSO). We compared these methods to each other in order to come to a better understanding of how they perform on a given problem, their strengths and weaknesses. The way these regression models work, together with many other Machine Learning algorithms is that they estimate relationships between the so-called response variable and a set of predictors. To do this, we need a set of data containing information of the system we want to study. In this study, we use use data generated from the Franke function eq. 10 with added white noise, then perform regression analysis with our three methods. The goal is to tweak the corresponding parameters for each regression algorithms so that it yields reliable results. After that, we will move on to a more practical example, where we will do the same analysis on topographical terrain data. Throughout the process we will be utlilizing resampling techniques such as *k-fold Cross Validation* and *Bootstrap* to tune the parameters and assess out models, while comparing the error and the runtime. Further, the theory behind the Bias-Variance tradeoff is applied to show that the Mean Square Error (MSE) can be broken down into an irreducible error together with the squared bias and the variance.

II. THEORY

i. Linear Regression

Given a set of outcomes denoted $\mathbf{y}$, where $\mathbf{y}$ is vector of size n also referred to as *the response variable*, and a set of inputs $\mathbf{X}$ with p characteristics stored in a matrix of the dimensions $\mathbb{R}^{n\times p}$. Where the matrix $\mathbf{X}$ is also referred to as the *design matrix*. A linear regression model utilizes the correlation between the response variables $\mathbf{y}$ and the predictors $\mathbf{X}$ in order to construct a *function* $\mathbf{y(X)}$ under the assumption that the set could be linearly fitted. Then the response as a function of the predictors could be written as:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{1}$$

where ϵ accounts for the error in the linear model $\mathbf{X}\beta$ wrt. the response $\mathbf{y}$. The parameter vector β contains the linear coefficients β_i, where β_i are the unknown variables we seek out to solve in a regression problem.

Generally the error terms ϵ_i in ϵ are non-zero. We regard our linear model as the fitted response, denoted $\widetilde{\mathbf{y}}$, wrt. the observed variables y_i in $\mathbf{y}$.

$$\begin{aligned}\widetilde{\mathbf{y}} &= \mathbf{X}\beta \\ &= \mathbf{y} - \epsilon\end{aligned} \tag{2}$$

Ideally we want to be able to determine β with the least error ϵ possible. Which in return would yield the best possible linear fit of the response $\mathbf{y}$.

ii. Ordinary Least Squares

The ordinary least squares method minimizes ϵ by introducing *a cost function*, denoted $C(\beta)$, which in turns measures the spread between the observed values y_i and the corresponding terms $\widetilde{y_i}$ in the approximated parameter. Our cost function is the mean squared error (MSE), *[1] p. 29.*

$$\begin{aligned}MSE(y_i,\widetilde{y_i}) &= \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\widetilde{y_i})^2 \\ &= \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i-\beta_0+\sum_{j=1}^{p}\mathbf{X}_{ip}\beta_p\right)^2\end{aligned} \tag{3}$$

then the cost function in matrix form is expressed in the terms:

$$C(\beta) = \frac{1}{n}(\mathbf{y}-\mathbf{X}\beta)^T(\mathbf{y}-\mathbf{X}\beta), \tag{4}$$

The optimal value for β is computed by minimizing the spread of $C(\beta)$, this means we intend to solve the problem:

$$\beta_{optimal} = \frac{1}{n}\min_{\beta\in\mathbb{R}^p}\left\{(\mathbf{y}-\mathbf{X}\beta)^T(\mathbf{y}-\mathbf{X}\beta)\right\},$$

this is done by requiring the first derivative wrt. β to be zero, since this condition coincides with the least deviation from the response variables y_i. Following the same procedure that is further detailed in *[2]*, we find that

$$\begin{aligned}\frac{\partial C(\beta)}{\partial\beta} &= -\frac{2}{n}\mathbf{X}^T(\mathbf{y}-\mathbf{X}\beta) = 0 \\ \Rightarrow\quad \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\beta \\ \beta_{optimal} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

Even though $\mathbf{X}$ is an $n\times p$ matrix, the product $\mathbf{X}^T\mathbf{X}$ is always a square matrix of the dimensions $p\times p$. Given the fact that - for the most part - $n \gg p$, computing the inverse of $\mathbf{X}^T\mathbf{X}$ is considered a mundane task and shouldn't be a problem.

So now we can calculate the *prediction model* by simply $\widetilde{\mathbf{y}} = \mathbf{X}\beta_{optimal}$, all derived from the MSE-method for establishing a cost function.

The Design Matrix in 2 Dimensions

In theory a design matrix possesses any set of linearly independent functions of the predictors. But for all intends and purposes we're focusing on a 2 dimensional polynomial expansion. For a design matrix $\mathbf{X}$ containing two variables x and y constructing the linear mapping of our response variable, the elements are set up as follows

$$x^p,\ x^{p-1}y,\ x^{p-2}y^2,\ \ldots,\ x^2y^{p-2},\ xy^{p-1},\ y^p$$

If we were to construct a matrix with a polynomial of degree $p = 3$, The set of degrees shaping up our polynomial is distributed as shown in Table I.

Table I. Table containing the components that make up a polynomial of degree $p = 3$.

$p = 3$	x^3	x^2y	xy^2	y^3
$p = 2$	x^2	xy	y^2	
$p = 1$	x	y		
$p = 0$	1			

In other words the permutation of each degree from p to 0 is as follows $x^i y^{p-i}$ along each row. Where $p = 0$ coincides with the intercept β_0. So as an example an observed element $\widetilde{y}_i$ expressed as a 2nd degree polynomial becomes

$$\widetilde{y}_i = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 xy + \beta_5 y^2$$

iii. Ridge and Lasso Regression

OLS is done by utilizing the L^2-norm, $\|\mathbf{x}\|_2 = \sqrt{\sum x_i^2}$ as a metric for error which is a direct result of MSE:

$$\begin{aligned} C_{OLS}(\beta) &= \|y - \widetilde{y}\|_2^2 = \|y - \mathbf{X}\beta\|_2^2 \\ &= (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \end{aligned}$$

Which in turns could carry coefficients with low bias and a large variance (more on that later on!), while it treats all parameters in the same manner without taken into account that some predictors may be more important thant others.

Ridge regression on the other hand introduces a *regularization parameter* $\lambda \geq 0$ in the cost function, *[2] p. 64*, it's solely there to penalize large β values. This regularization parameter helps resolve a big issue which we encounter with OLS, and that is that $(\mathbf{X}^T\mathbf{X})^{-1}$ is almost never invertible, which gives infinitely many solutions.

$$\begin{aligned} C_R(\beta) &= \|\mathbf{y} - \widetilde{\mathbf{y}}\|_2^2 + \lambda\|\beta\|_2^2 \\ &= (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta \end{aligned}$$

A side note $\lambda = 0$ reverts us back to OLS. By taking the derivative wrt. β, similar to OLS, we obtain the optimal values for the coefficients β, when enforcing $C_R(\beta) = 0$. Which in turn yields

$$\begin{aligned} \frac{\partial C_R(\beta)}{\partial \beta} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) + 2\lambda\beta = 0 \\ \Rightarrow \quad \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\beta + \lambda\beta \\ \beta_{optimal} &= \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

with $\mathbf{I}$ being a $p \times p$ identity matrix.

iv. Lasso Regression

By replacing the L^2-norm in the Ridge regression for the regularization parameter with an L^1-norm, $\|x\|_1 = \sum |x_i|$, we then obtain another shrinkage method called the Lasso (Least Absolute Shrinkage and Selection Operator) method, with a cost function, *[1] p. 219.*

$$\begin{aligned} C_L(\beta) &= \|\mathbf{y} - \widetilde{\mathbf{y}}\|_2^2 + \lambda\|\beta\|_1 \\ &= (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda\sqrt{\beta^T\beta} \end{aligned}$$

Its advantage over OLS and Ridge, is that Lasso can preform *variable selection* in the sense that some coefficients β_i will be set to zero as a result of the minimization.

v. Error Analysis

We chose to assess the models accuracy using different error indicators. Primarily with the help of the *mean square error* MSE (3). In addition to the *mean absolute error*:

$$MAE(y, \widetilde{y}) = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - \widetilde{y}_i| \tag{5}$$

Where for an ideal case both error indicators should be ≈ 0.

Another measure is the R^2-score, *[1] p. 70.* Giving by

$$R^2(y, \widetilde{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \widetilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{6}$$

where $\bar{y}$ is the mean value of y

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$$

The value for the R2 score is ideally between 0 and 1, where a value of 1 indicates a strong correlation between the response y_i and the approximation $\widetilde{y}_i$.

vi. Bias-Variance Trade-off

To understand the *bias-variance trade-off* we start by looking at an arbitrary case where we consider the dataset $\mathcal{L}$ consisting of the data $\mathbf{X}_{\mathcal{L}} = \{(y_j, \mathbf{x}_j),\ j = 0, 1, \ldots n-1\}$. Our first assumption is that the output data is generated from a noisy model

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

Here ϵ is a normally distributed with mean $\mu = 0$ and standard deviation σ.

The OLS regression method computes an approximation of f in terms of the coefficients β and the design matrix $\mathbf{X}$, $\widetilde{\mathbf{y}} = \mathbf{X}\beta$. It does this by *minimizing* the cost function which for this case is the MSE.

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

You can find the derivation in IV Appendix, result

$$\begin{aligned}\mathbb{E}\left[(\mathbf{y} - \widetilde{\mathbf{y}})^2\right] &= \frac{1}{n}\sum_i (f_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2 \\ &+ \frac{1}{n}\sum_i (\widetilde{y}_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2 + \sigma^2 \\ &= \underbrace{\mathbf{Bias}^2(\widetilde{\mathbf{y}}) + \mathbf{Var}(\widetilde{\mathbf{y}})}_{\text{Mean Square Error}} + \sigma^2\end{aligned} \tag{7}$$

This is called the *test* or *generalization error*. The last term σ^2 gives us the irreducible error [2] p.37.

vii. Resampling Methods

The goal of resampling is to increase the statistical significance of our regression analysis and the models reliability. There's many approaches to achieve just that and they are particularly useful when we are working with small datasets. In this project we will look at *bootstrap* and *cross-validation* and comparing them to one another.

Bootstrap

The bootstrap method randomly draws points from the original dataset, creating a smaller dateset to work with. The idea is to sample/draw random points from the original dataset B times and for each bootstrapping set we compute the variables we're interested in. In our case we focused mainly on calculating the MSE, bias and variance. Then we compute the mean value over all the bootstrapping sets for each of the variables.

Cross-validation

The idea of cross-validation is to estimate the the *expected test error*

$$\text{Err} = \mathbb{E}[L_2(y, \hat{y})]$$

of our model. The error is assesed by splitting the dataset into k equally sized folds as shown in Figure 1. Right after the dataset has been shuffled around. We do that to ensure that the k datasets are randomly drawn out. No need to do so if you know beforehand that your dataset is randomly set up.

After splitting the dataset we then pick one of the folds as our test data and utilize the rest of the sets as our training data and cross-validate the model with the test data. We do that k times looping through each fold. For each iteration we calculate the R^2 score, MSE and any other statistic we're interested in for that matter. Finally we compute the mean value over all cross-validations for each of the variables.

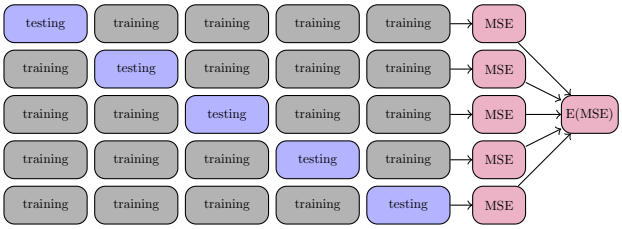

Figure 1. An illustration of the idea behind a k-fold cross-validation. In this figure we see an example of a 5-fold cross-validation.

III. RESULTS & DISCUSSIONS

In order to assess how well any of the regression models preforms, we will first do a test phase on data generated by the Franke Function eq. 10. We will then lastly apply our regression algorithms to a more real-life problem on geographical data.

i. Franke's function

The Franke Function is defined as

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \tag{8}$$

$$+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right)$$

$$+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \tag{9}$$

$$- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \tag{10}$$

where $(x,y) \in [0,1] \times [0,1]$.

In order to evaluate the behaviour of our fit, we will also add a stochastic noise $\epsilon \sim N(0,\sigma^2)$ to our Franke-function

$$z = f(x,y) + \epsilon$$

This stochastic noise will ensure us that our function is more alike our geographical data. Hence, it's more representative of the geographical data we're going to be working with later on.

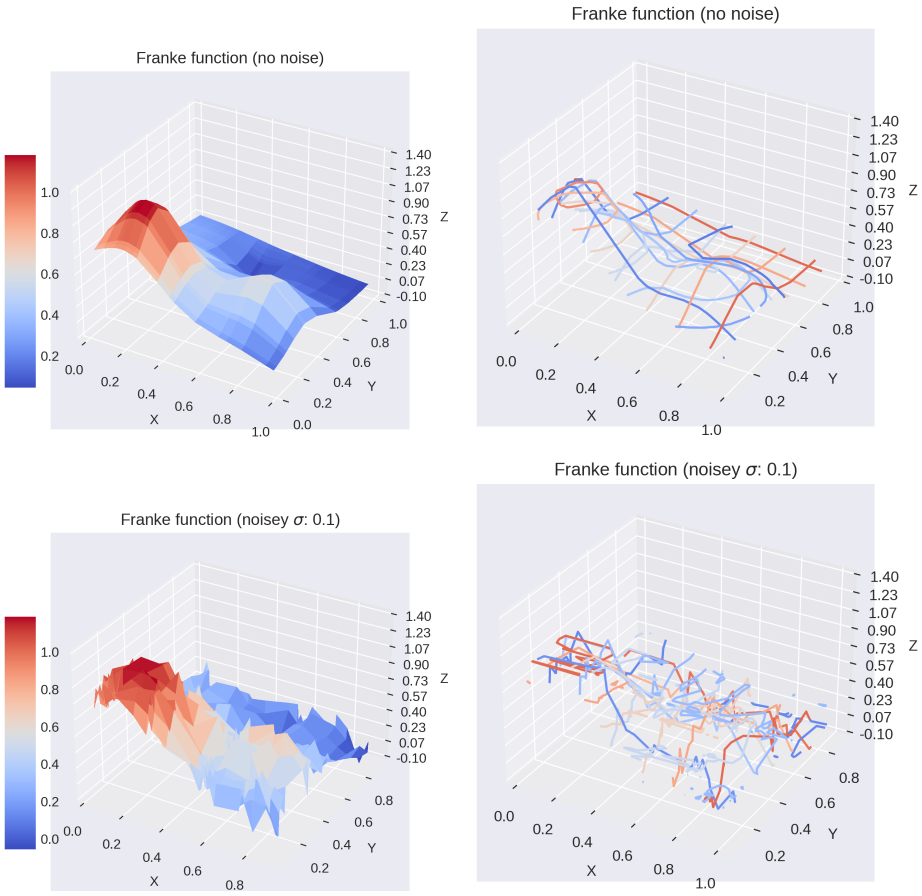

Figure 2. Franke-function and Franke + ϵ: 3D-plot (left) contour (right).

ii. Exercise 1

We assessed the confidence interval for the estimators β by computing $\mathrm{Var}(\beta) = \sigma^2 \left(\mathbf{X}^T\mathbf{X}\right)^{-1}$, ch. 5.2 in [3]. σ^2 is obtained by computing the variance of the noise factor in franke function, in other words $\mathrm{Var}(\epsilon) = \sigma^2$

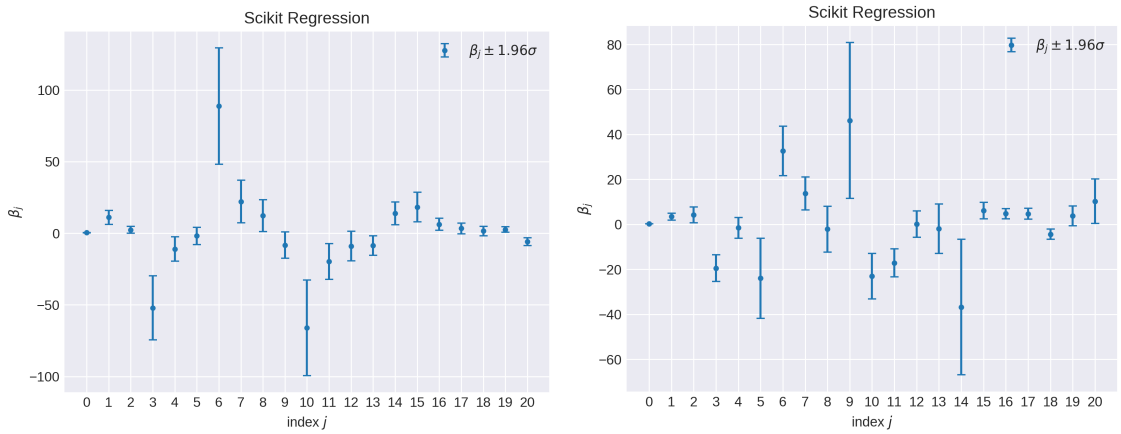

Figure 3. Finding the confidence intervals by first scaling the data using `Numpy_scale_split` lhs. and `Scikit_scale_split` rhs. OLS-regression from `Scikit` is used to generate the data.

The data generated here if for regression model with a polynomial degree $p = 5$. The training data makes up 4/5 of the entire data set. As for the error-analysis we chose two approaches, calculating the error manually using the module `error.py` and using `Scikit`'s own MSE and R^2-score.

Table II. MSE and R2-score for both `Scikit` and `error.py`

Deg	2		3		5	
Error Type	R^2	MSE	R^2	MSE	R^2	MSE
Train Error	0.6683	0.0237	0.8661	0.0157	0.8870	0.0107
Test Error	0.6214	0.0196	0.7846	0.0228	0.7492	0.0194

Both modules generated the same values for each of the polynomial degrees $p = 2, 3, 5$. The data was scaled using `Scikit`'s StandardScaler, which scales the data by computing

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}$$

where $\hat{\mathbf{x}}$ is the scaled output, μ is the mean of the data $\mathbf{x}$ to be scaled and σ is the standard deviation.

There is no need for us to scale our data but it's good practice in ML nonetheless. In real world applications, different features have different units and numerical scales. Thus, in order to avoid poor preforming learning methods yielding unreliable results, the data is to be scaled minimizing the impact of any outliers. Although it's worth noting that this form of scaling has a drawback in that it doesn't ensure a particular maximum or minimum, ch. 3.8 in [3].

iii. Exercise 2

We made an MSE-analysis using the bootstrap resampling technique to find how the test error diverge from the training error as p increases, the results are shown in fig. 4.

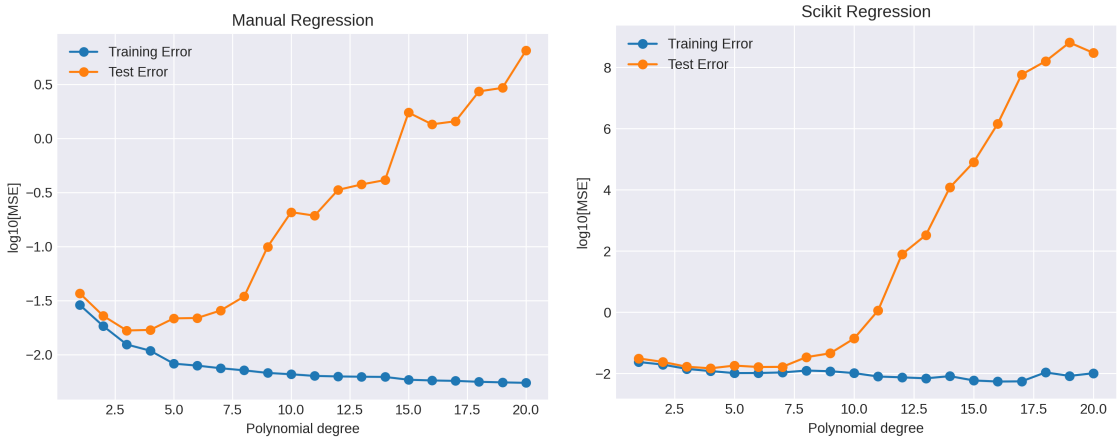

Figure 4. Train vs. Test MSE resembling Fig. 2.11 in [2]. Scaling and OLS using `Numpy` LHS `Scikit` RHS, $N = 20$.

The MSE for both `Manual` and `Scikit` regression follows the same trend but the error is significantly lower for Manual-OLS. We've also done a bias-variance tradeoff analysis following what we have found in eq. 7 to confirm $\texttt{Error} \geq \texttt{Bias} + \texttt{Variance}$.

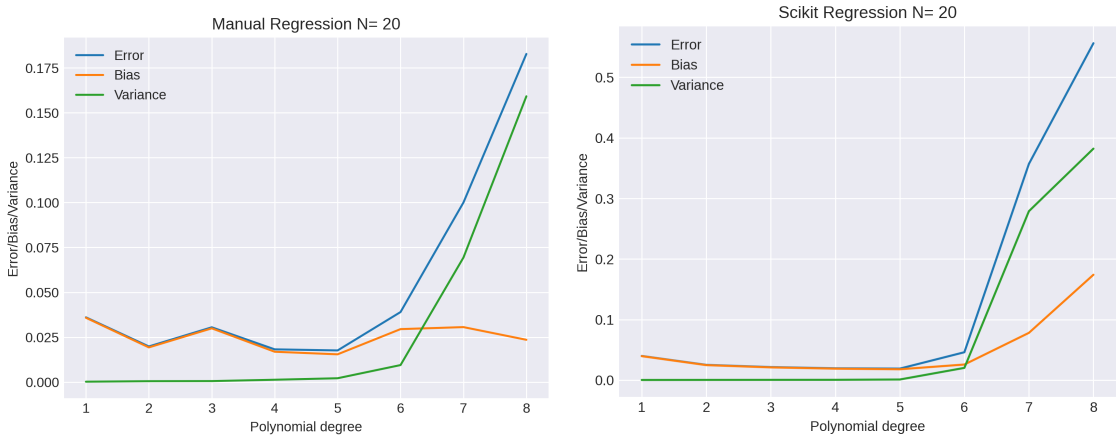

Figure 5. Bias-Variance tradeoff of the Franke function with $N \times N$ mesh points. Scaling and OLS using `Numpy` LHS `Scikit` RHS.

Agreeing with our findings for the MSE-analysis, both graphs more or less follow the same trend where again the Manual-OLS error is significantly lower.

iv. Exercise 3

For the cross-validation we had to approaches one utilizing `sklearn.model_selection.KFold` to split the data in to k number of folds before doing cross-validation. And one utilizing `sklearn.model_selection.cross_val_score` which is `Scikit`'s cross-validation. Tab. III below shows the results we've obtained using these two approaches.

Table III. Average MSE as a function of polynomial degree p for the cross-validation resampling technique.

Deg	2		3		5	
Error Type	Scikit_CV	scores_KFold	Scikit_CV	scores_KFold	Scikit_CV	scores_KFold
Train Error	0.1328	0.1328	0.0363	0.0363	0.1353	0.1353
Test Error	0.6214	0.0196	0.7846	0.0228	0.7492	0.0194

We compared the MSE from a cross-validation approach against the bootstrap MSE for the test error as shown in fig. 4.

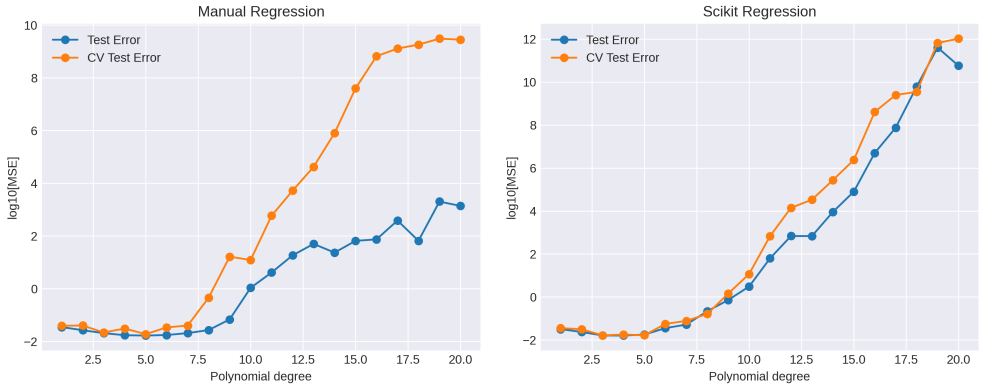

Figure 6. KFold Cross-validation done for $k = 10$ folds plotted against the corresponding bootstrap MSE, for both manual and `Scikit` regression

The MSE of bootstrap for the manual regression is by far the lowest. As for the cross-validation MSE it diverges from the bootstrap MSE of the manual regression but follows the bootstrap MSE of the `Scikit` regression. It's worth noting that the CV MSE for `Scikit` is 2 orders of magnitude higher than it's manual counterpart. In addition we measured the time elapsed for each of the algorithms to see which computationally preforms the best.

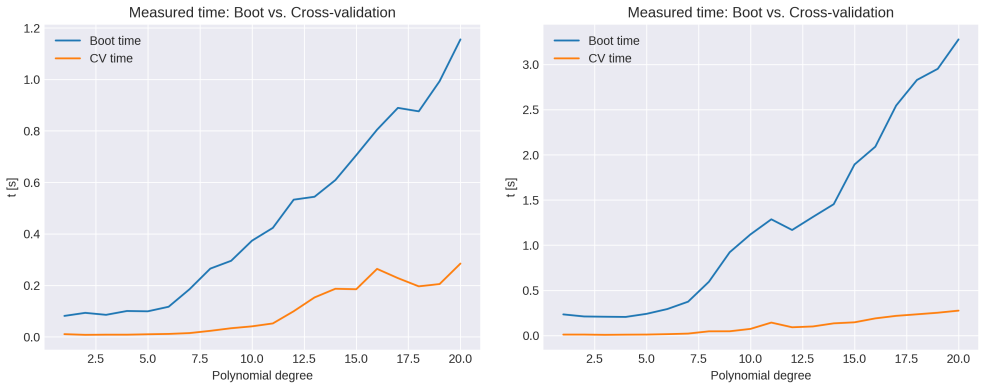

Figure 7. Time elapsed KFold Cross-validation vs. Bootstrap for both manual (left) and `Scikit` (right) regression.

As shown in fig. 7 the CV is noticeably better in terms of computation and it's worth noting that these two graphs also show that doing the regression manually results in a considerable boost in performance.

v. Exercise 4

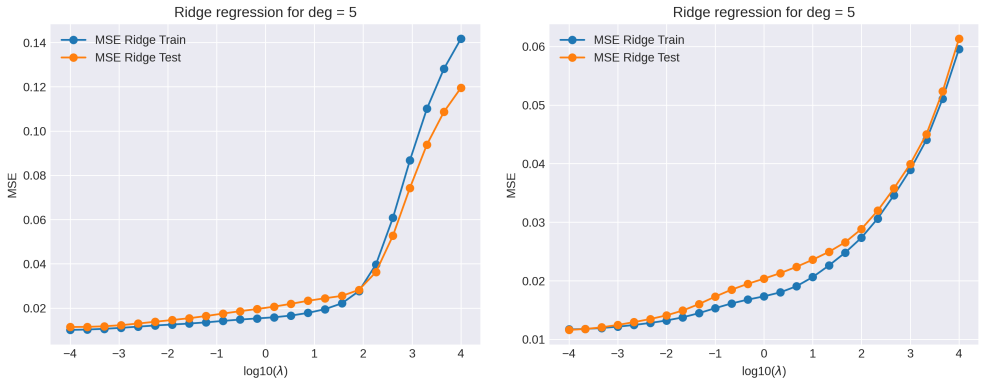

Figure 8. Train vs. Test MSE for Ridge regression of a polynomial degree $p = 5$ as a fucntion of λ, Manual (LHS) `Scikit` (RHS).

For Ridge we made the same MSE-analysis as for OLS, this time for a range of different λ from 10^{-4} to 10^4. The figures below are picked out from fig. 23 in **IV. Appendix i**

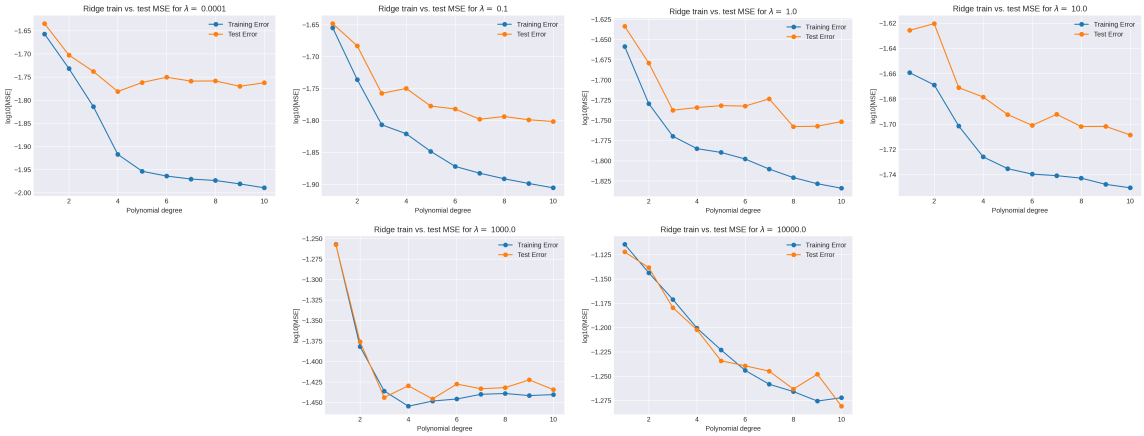

Figure 9. Ridge Train vs. Test MSE for a range of $\lambda \in [10^{-4}, 10^4]$

The plots in fig. 9 show how Ridge regression converts back to OLS as $\lambda \to 0$. As for the MSE it operates within the threshold of OLS for the same polynomial degree $p = 10$.

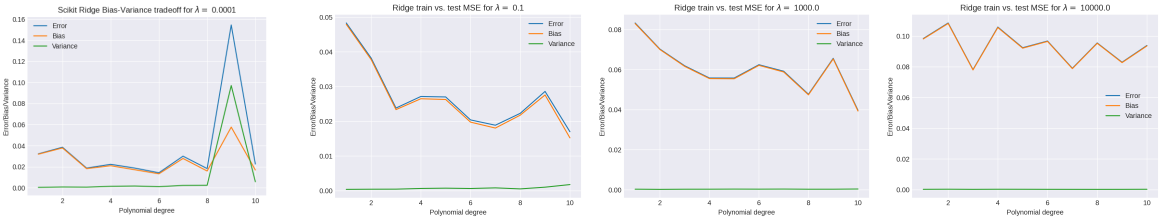

Figure 10. The bias-variance tradeoff for the same range of λ

Fig. 10 agrees with what we found in fig. 5 in that Ridge converts back to OLS as $\lambda \to 0$. We can clearly see that the variance overtakes the bias at $p = 8$, which is very similar to fig. 5.

Figure 11. Ridge KFold Cross-validation for k= 10 plotted against corresponding bootstrap MSE, for the same range of λ.

For low values of λ the MSE values for both the bootstrap and the cross-validation are within a certain threshold of each other, which expected as Ridge converts back to OLS as $\lambda \to 0$. For higher values of lambda the CV MSE diverges, fig. 29 in **IV. Appendix i**, paints a better picture as to what happens in the interval $p \in [5, 20]$. For a relatively low λ the MSE increases with many orders of magnitudes from $p \approx 5$. As for a relatively large λ the model becomse steadily more stable especially at a point $p \approx 15$. But that's solely due to the fact that large values of λ zeroes out almost every estimator as we will see later on.

vi. Exercise 5

We start by plotting the values of the estimators for a range of λ for $p = 4$

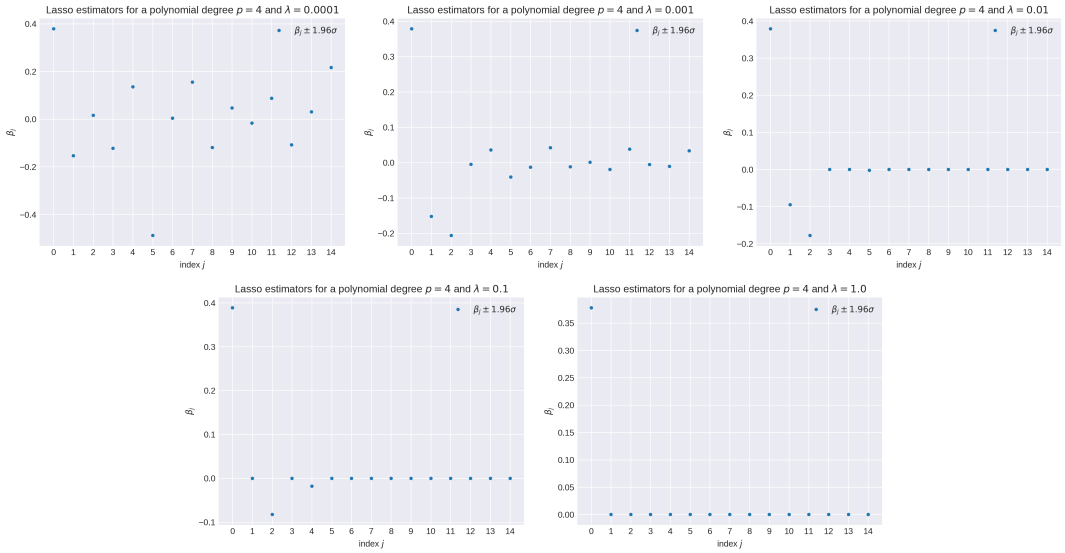

Figure 12. Lasso estimators graph for a range of λ, showing the value of each β. Effectively hinting the shape of the model.

We observe that for $\lambda = 1$ Lasso zeroes out any of the estimators except for β_1, meaning that our model follows a flat plane which is not the case for the franke function. Further more fig. 26 shows that this trends continues for higher λ.

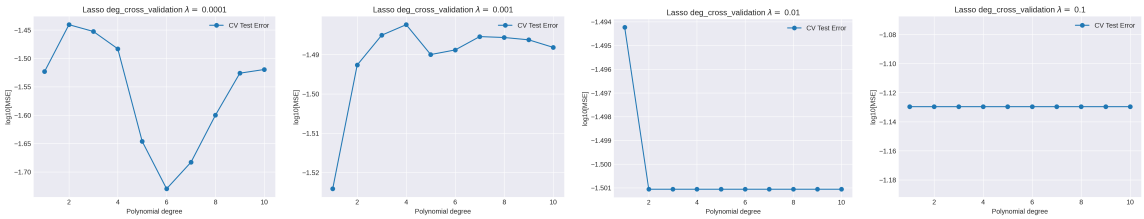

Figure 13. Lasso KFold cross-validation for k= 10, for a range of λ.

For the CV the lowest MSE is achieved for $\lambda < 0.01$. The graph for $\lambda = 0.01$ in fig. 13 although it gives low MSE values it does not behave in a normal way for a regression model, same for $\lambda \geq 0.1$. On the other hand $\lambda = 1\text{e}{-3},\ 1\text{e}{-4}$ behave within expectations, $\lambda = 1\text{e}{-4}$ being the best of the two.

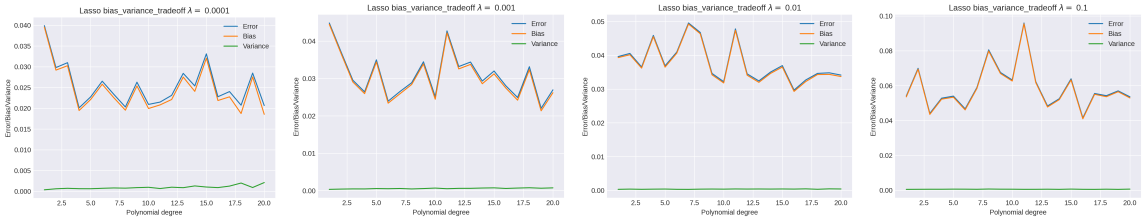

Figure 14. Lasso bias-variance tradeoff as a function of polynomial degree p, for a range of λ.

The bias-variance tradeoff behaves similarly to that of Ridge in fig. 10. What we mean is that the error in both cases is solely made up from the bias for higher λ. $\lambda = 1\text{e}{-3},\ 1\text{e}{-4}$ gives a stronger leeway for the variance although it's not much.

Comparing both Ridge and Lasso for $\lambda = 1\text{e}{-3}$:

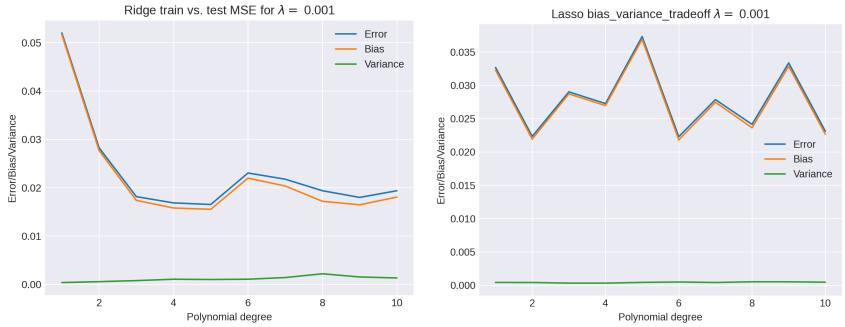

Figure 15. The bias-variance tradeoff of Ridge(Left) and Lasso (Right).

The total error of Ridge is lower than that of Lasso, especially in the interval $p \in [3, 5]$; $p = 5$ being the best. Looking at fig. 24 and 27 for $\lambda = 0.001$, plus adding a CV vs. Boot analysis for the OLS to compare all 3 methods.

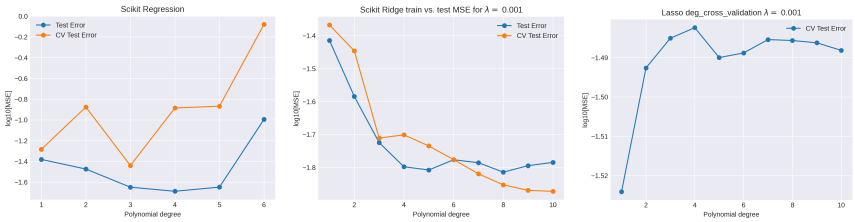

Figure 16. 1st OLS CV vs. test error, 2nd Ridge CV and test error, 3rd Lasso CV.

Looking at the graphs in fig. 16 for $p \in [3, 6]$: Ridge has the lowest MSE of the three hovering between $10^{-1.8} - 10^{-1.7}$, while Lasso well exceeds that for this interval. As for OLS the MSE is somewhere around $10^{-1.4}$ at it's best for $p = 3$.

vii. Exercise 6

We revisit the methods we've worked with until now to assess a real world problem. What you're seeing below is the analysis from 1-5 using the data SRTM_data_Norway_1.tif.

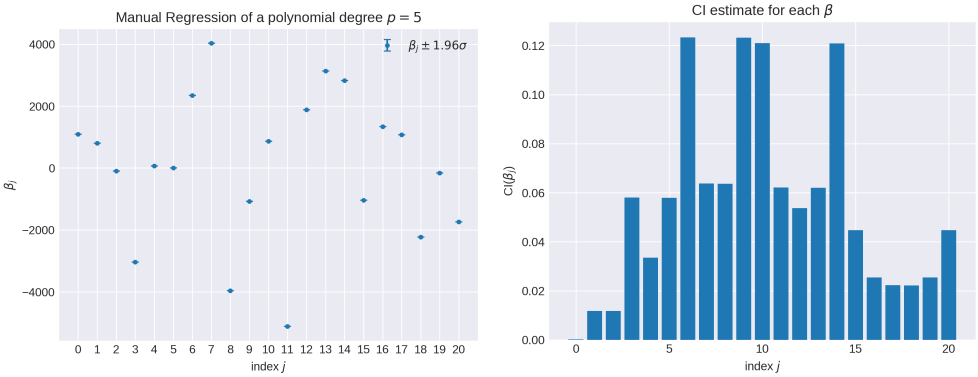

Figure 17. The confidence interval for the terrain data, (left) same plot as fig. 3. (Right) an errorbar-plot showing the CI estimates for each β_j.

From the errorbar-graph we see that there's some β (estimator) that stand out more than the others. Requiring the regression to be done using Ridge or Lasso will eliminate any overshooting estimators. But we still show the cross-validation results for OLS just as a reference point.

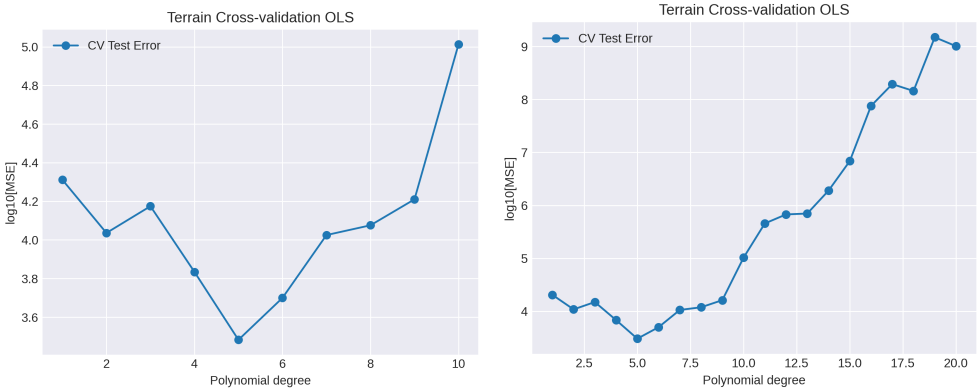

Figure 18. The cross-validation for the OLS as a function of p.

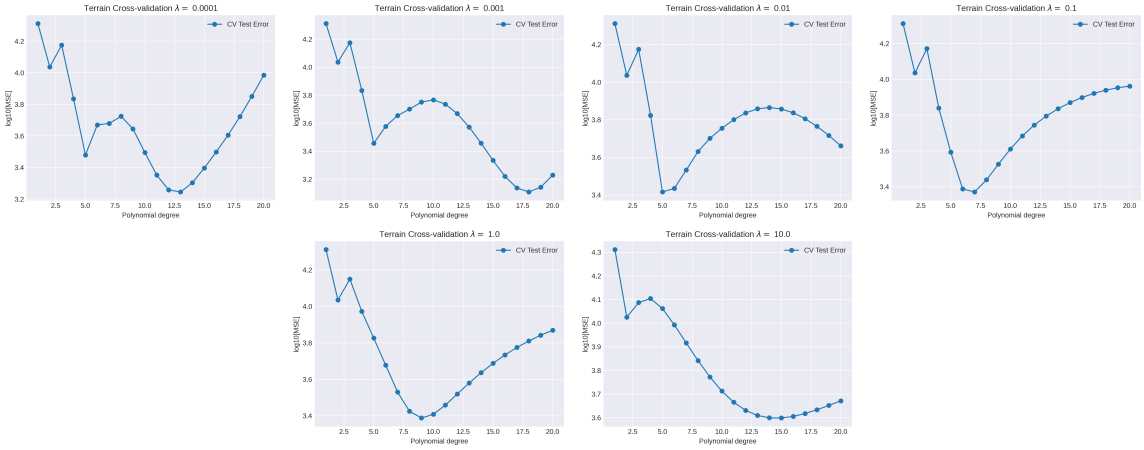

Figure 19. Ridge cross-validation for a range of λ as a function of the polynomial degree p

The best results are obtained by $\lambda \in [10^{-4}, 1]$ where the error always hovers under $10^{3.4}$. The λ with the best results by far is $\lambda = 0.001$ where the error hovers under $10^{3.2}$ for $p = [17, 19]$. But $p = 16$ should be taken under consideration as it's less complex and meets the threshold. As for $\lambda \geq 10$ the error from the cross-validation increases noticeably, as shown in fig. 29.

We know now, from our Ridge-analysis, that high values of λ are not reliable and zero out the estimators we're interested in. Thus, for our Lasso-analysis we compute the CV for up to $\lambda = 10$.

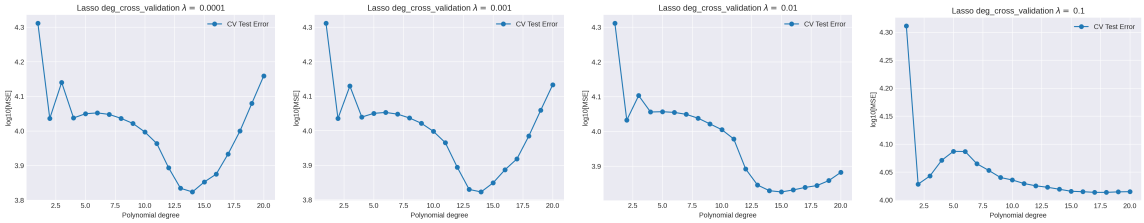

Figure 20. Terrain Lasso cross-validation up to the 20th polynomial degree for a range of λ.

As for Lasso the best results are obtained by $\lambda \in [10^{-4}, 10^{-2}]$, where the error hovers under $10^{3.9}$. Again, the λ that by far yielded the best results is $\lambda = 0.001$. The error hovers under $10^{3.9}$ for $p = [12, 16]$ with $p = 14$ being the most accurate of the bunch. Repeating what occurred for Ridge the threshold isn't met for higher λ, namely $\lambda \geq 0.1$. As shown in fig. 31.

Our findings

Ridge: best range $\lambda \in [10^{-4}, 1]$, the best is $\lambda = 0.001$ where the error hovers under $10^{3.2}$ for $p = [17, 19]$. But $p = 16$ should be taken under consideration

Lasso: best range $\lambda \in [10^{-4}, 10^{-2}]$, the best is $\lambda = 0.001$. for $p = [12, 16]$ with $p = 14$ being the most accurate

IV. APPENDIX

The Franke function:

$$
\begin{aligned}
f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right) \\
+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-7)^2 - (9y-7)^2\right)
\end{aligned} \tag{11}
$$

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

Our goal is to express the cost function as a sum of the bias and the variance of our model plus the variance that stems from the dataset itself, see (7). As discussed we assume that the data is generated from noisy model $\mathbf{y} = f(\mathbf{x}) + \epsilon$, in addition we add and subtract the expectation value of the estimator, denoted $\mathbb{E}\left[\widetilde{\mathbf{y}}\right]$.

$$
\begin{aligned}
\mathbb{E}\left[(\mathbf{y}-\widetilde{\mathbf{y}})^2\right] &= \mathbb{E}\left[(f + \epsilon - \widetilde{\mathbf{y}} + \mathbb{E}\left[\widetilde{\mathbf{y}}\right] - \mathbb{E}\left[\widetilde{\mathbf{y}}\right])^2\right] \\
&= \mathbb{E}\left[((f - \mathbb{E}\left[\widetilde{\mathbf{y}}\right]) + \epsilon + (\mathbb{E}\left[\widetilde{\mathbf{y}}\right] - \widetilde{\mathbf{y}}))^2\right]
\end{aligned}
$$

We then use the fact that the variance for both $\mathbf{y}$ and ϵ is equal to σ^2 and that the mean value of ϵ is by definition zero. In addition we know that both f and $\widetilde{\mathbf{y}}$ are not stochastic.Which then yields

$$
\begin{aligned}
\mathbb{E}\left[(\mathbb{E}[\widetilde{\mathbf{y}}] - \widetilde{\mathbf{y}})^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\widetilde{\mathbf{y}}\right]^2 - 2\mathbb{E}\left[\widetilde{\mathbf{y}}\right]\widetilde{\mathbf{y}} + \widetilde{\mathbf{y}}^2\right] \\
&= \mathbb{E}\left[\widetilde{\mathbf{y}}^2\right] - \mathbb{E}\left[\widetilde{\mathbf{y}}\right]^2 \\
&= \mathbf{Var}(\widetilde{\mathbf{y}}) = \frac{1}{n}\sum_i (\widetilde{y}_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2
\end{aligned}
$$

In addition the squared bias is defined as

$$
\mathbf{Bias}^2(\widetilde{\mathbf{y}}) = (f - \mathbb{E}[\widetilde{\mathbf{y}}])^2 = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2,
$$

which ultimately yields

$$
\begin{aligned}
\mathbb{E}\left[(\mathbf{y}-\widetilde{\mathbf{y}})^2\right] &= \frac{1}{n}\sum_i (f_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2 + \frac{1}{n}\sum_i (\widetilde{y}_i - \mathbb{E}[\widetilde{\mathbf{y}}])^2 + \sigma^2 \\
&= \mathbf{Bias}^2(\widetilde{\mathbf{y}}) + \mathbf{Var}(\widetilde{\mathbf{y}}) + \sigma^2
\end{aligned}
$$

i. Figures

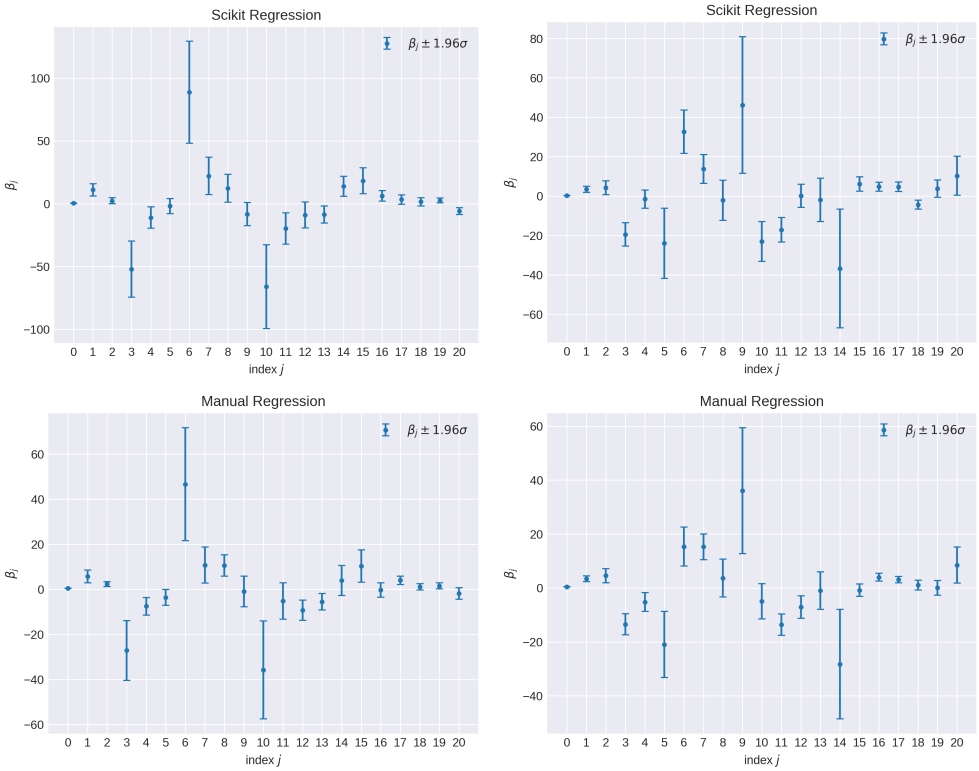

Figure 21. Scaling: `Numpy_scale_split` LHS `Scikit_scale_split` RHS. OLS-regression: `Scikit` upper row *Numpy* lower row.

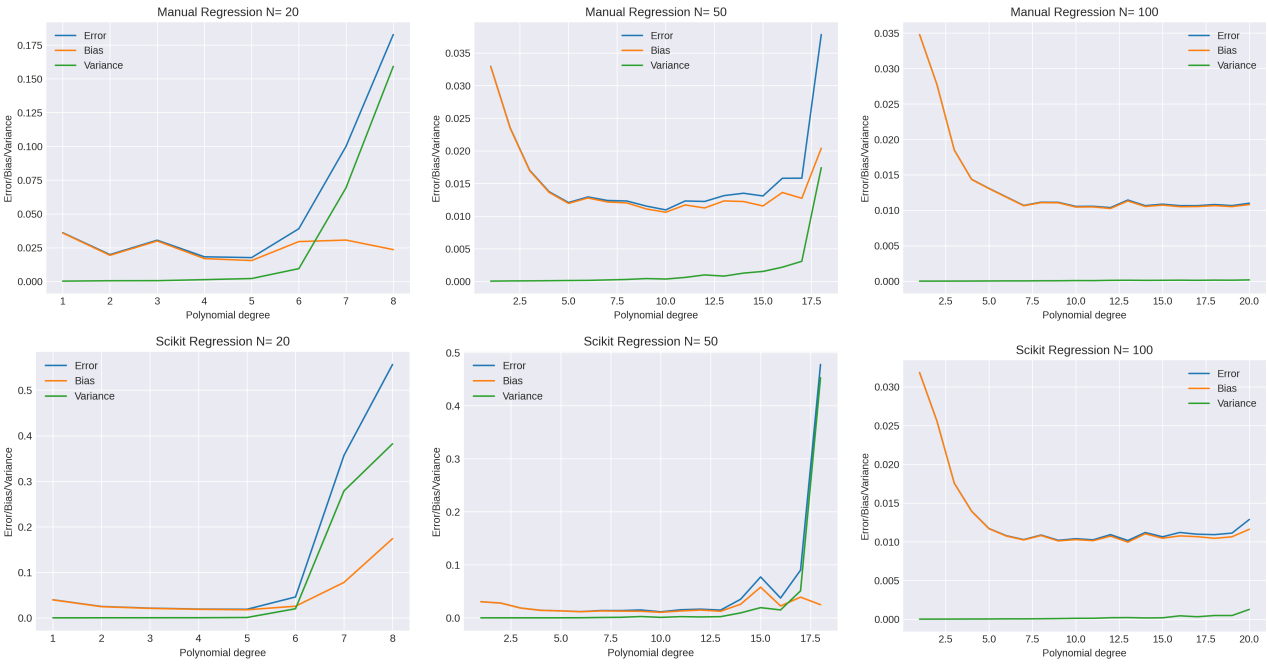

Figure 22. The bias-variance tradeoff scaling and OLS using `Numpy` upper row `Scikit` lower row.

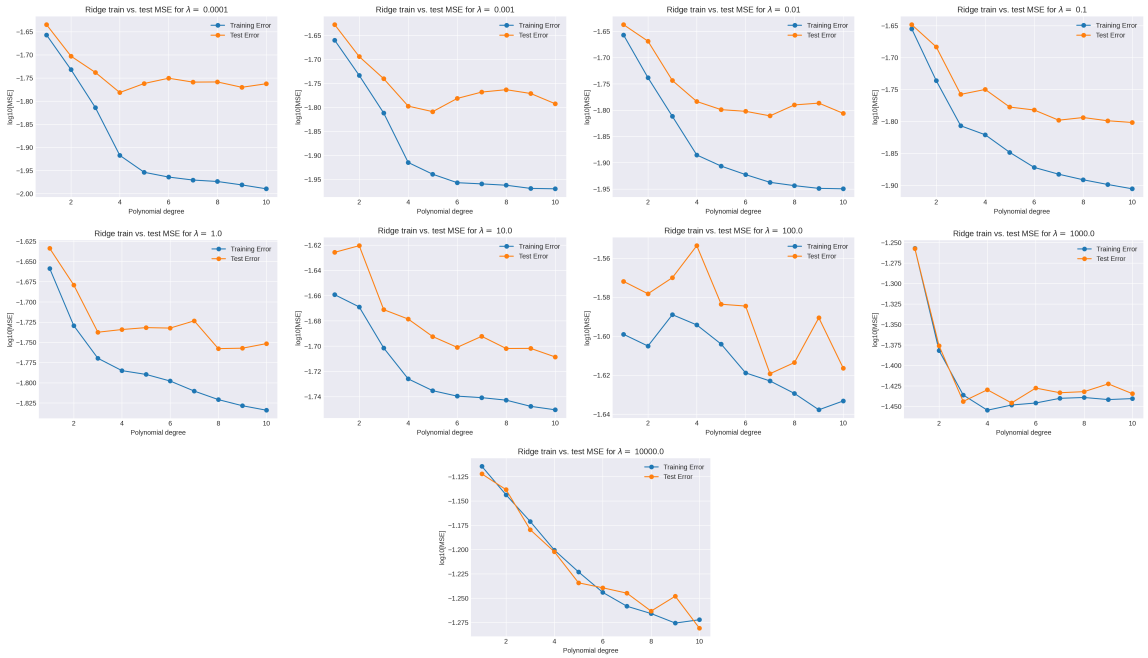

Figure 23. Ridge Train vs. Test MSE for a range of λ

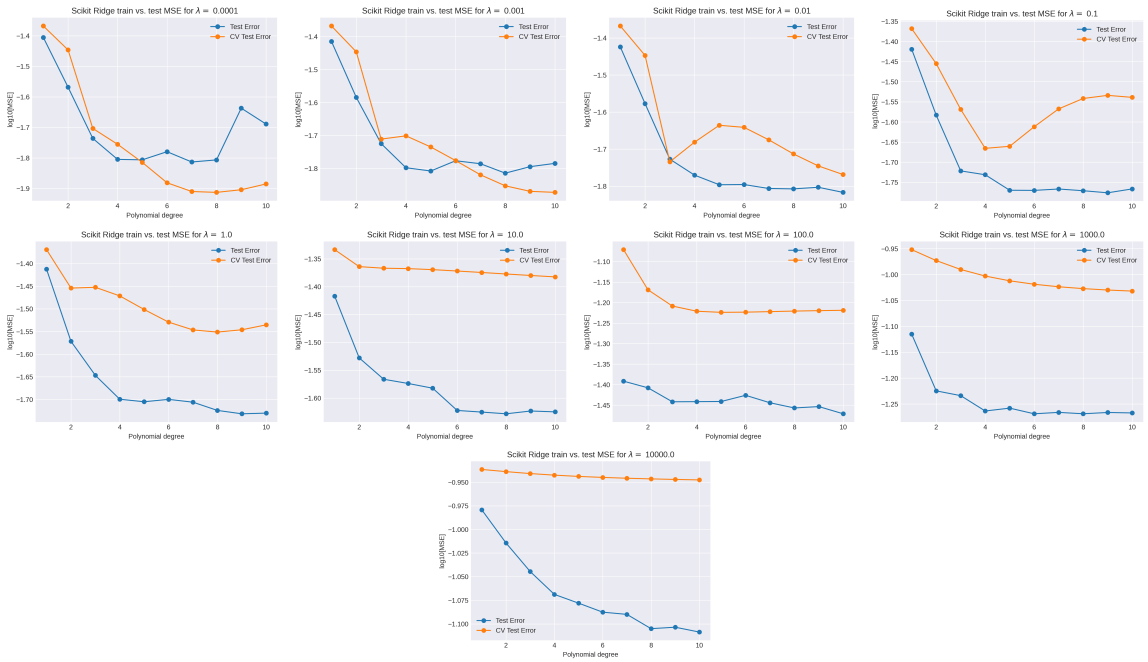

Figure 24. Ridge KFold Cross-validation for k= 10 plotted against corresponding bootstrap MSE, for a range of λ.

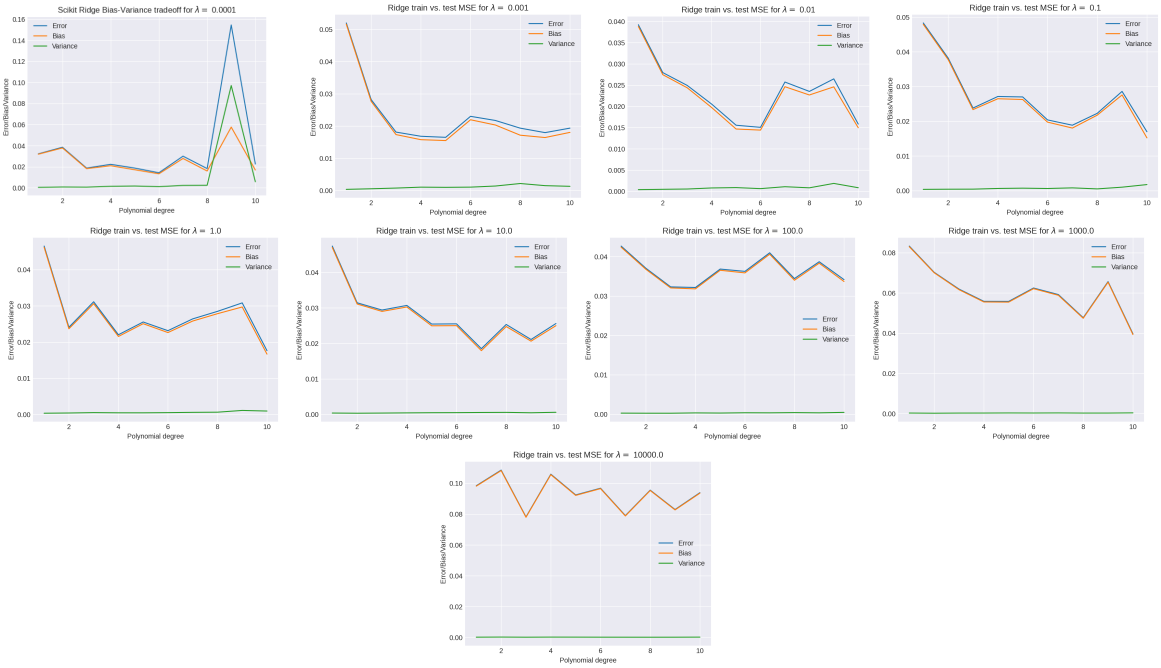

Figure 25. Ridge bias-variance tradeoff for a range of λ

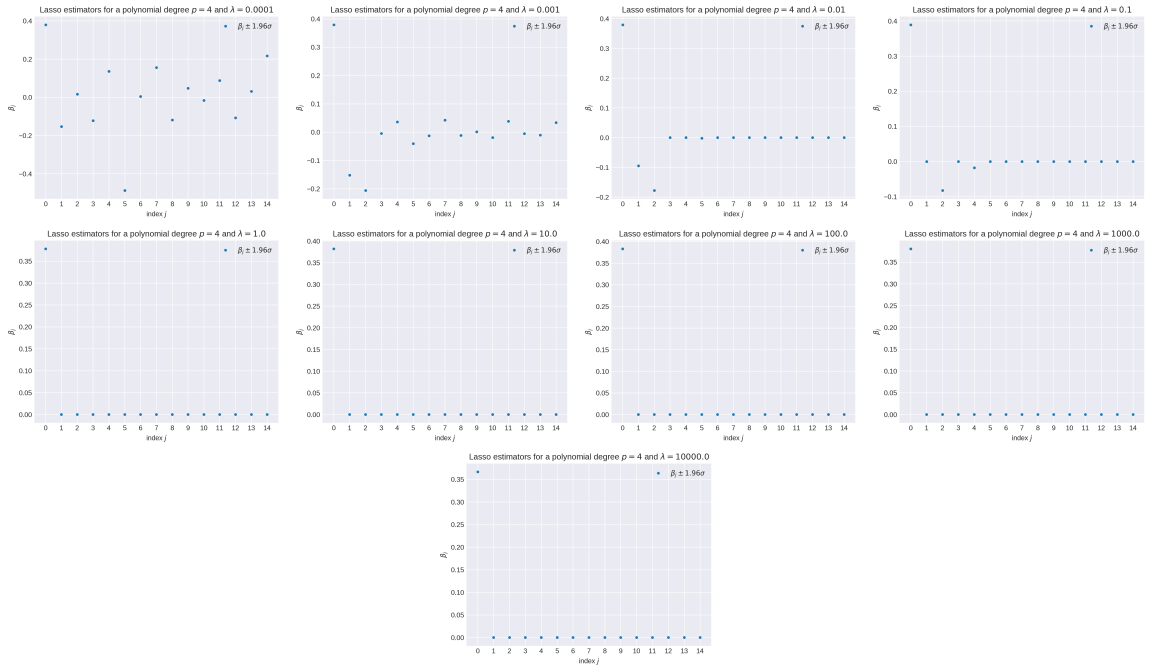

Figure 26. Lasso β (estimators) for a polynamial degree $p = 4$, for a range of λ.

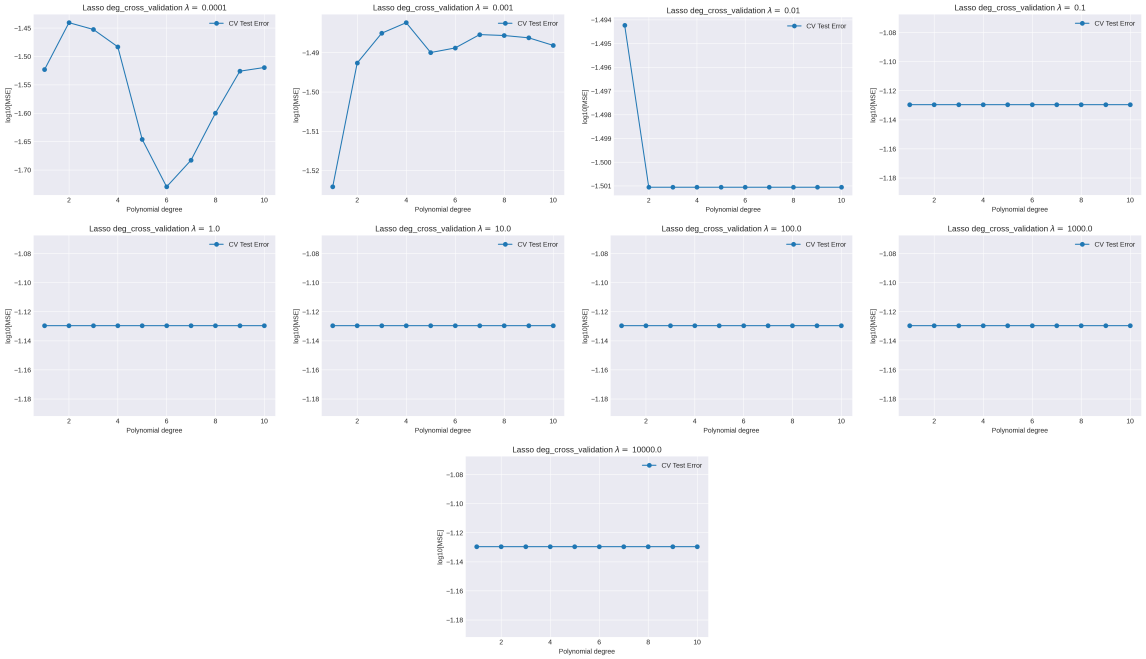

Figure 27. Lasso KFold cross-validation for k= 10, for a range of λ.

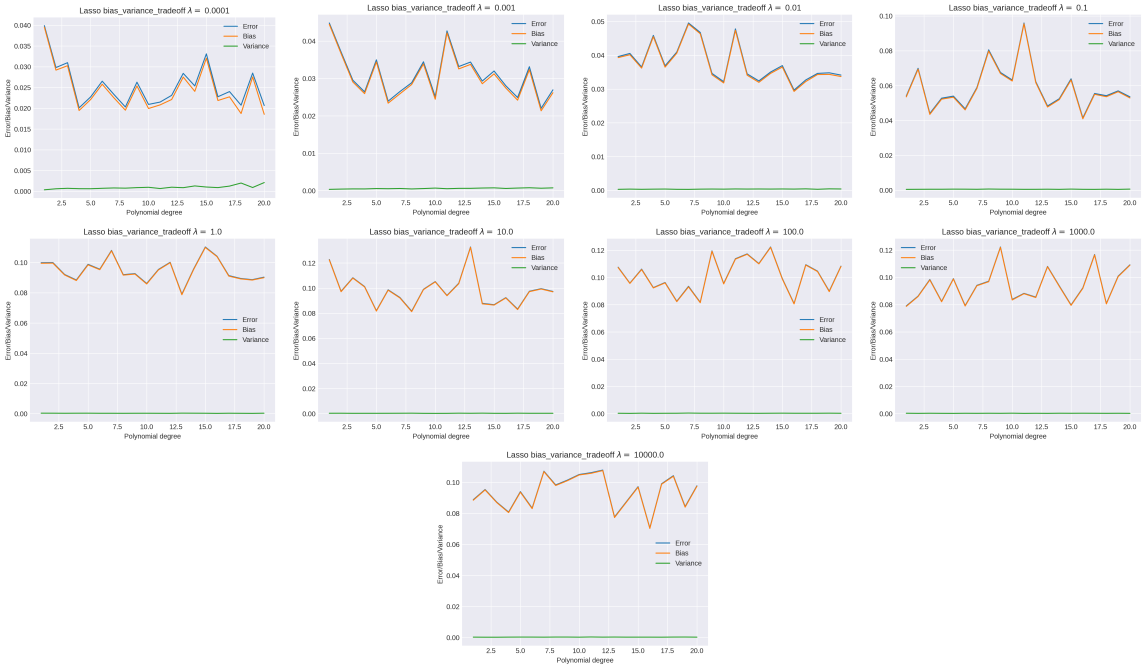

Figure 28. Lasso bias-variance tradeoff as a function of polynomial degree p.

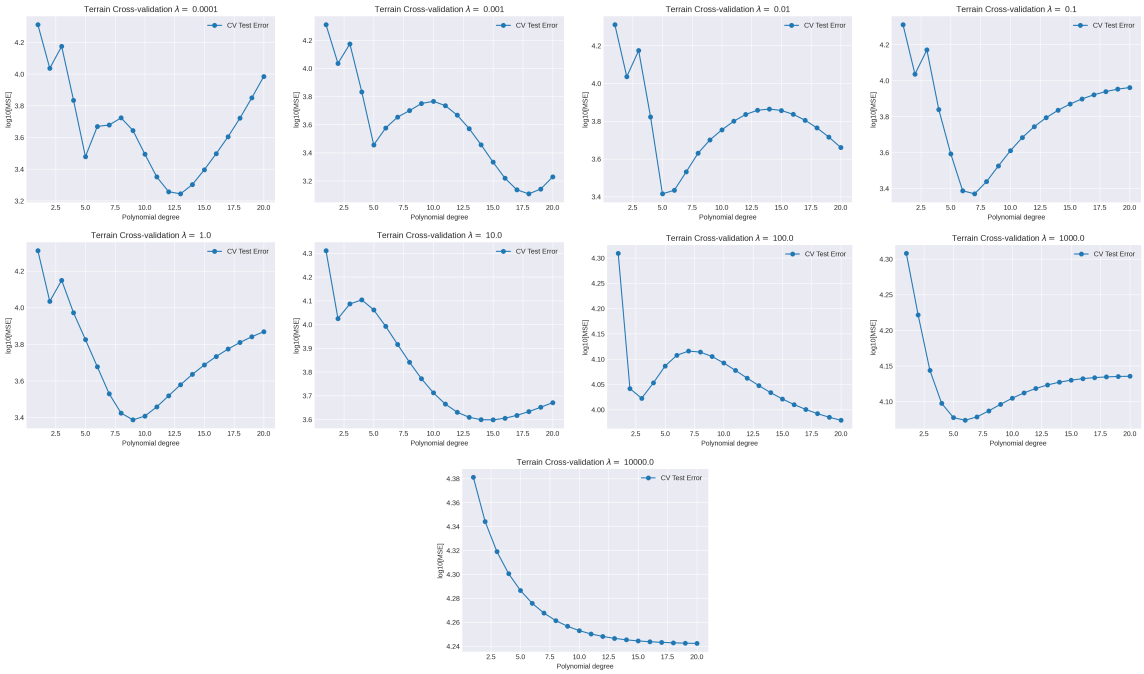

Figure 29. Terrain Ridge cross-validation up to the 20th polynomial degree for a range of λ.

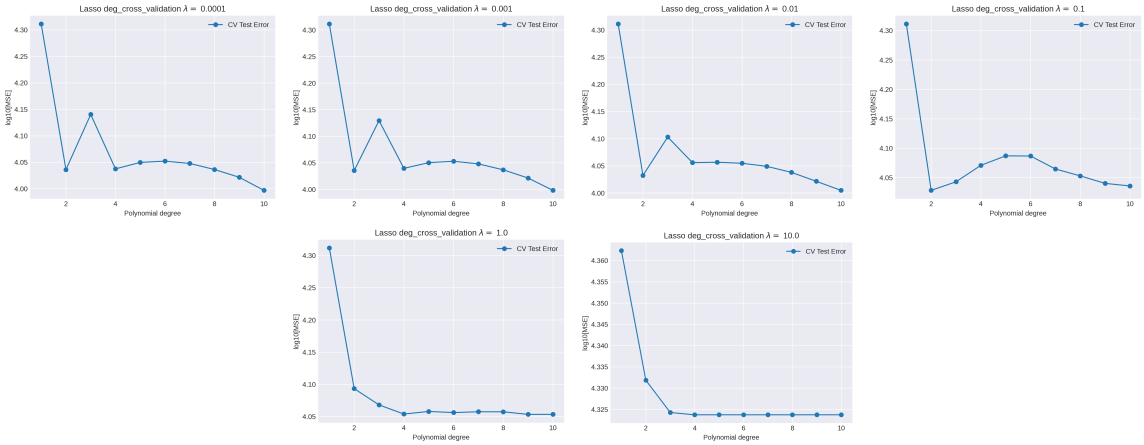

Figure 30. Terrain Lasso cross-validation up to the 10th polynomial degree for a range of λ.

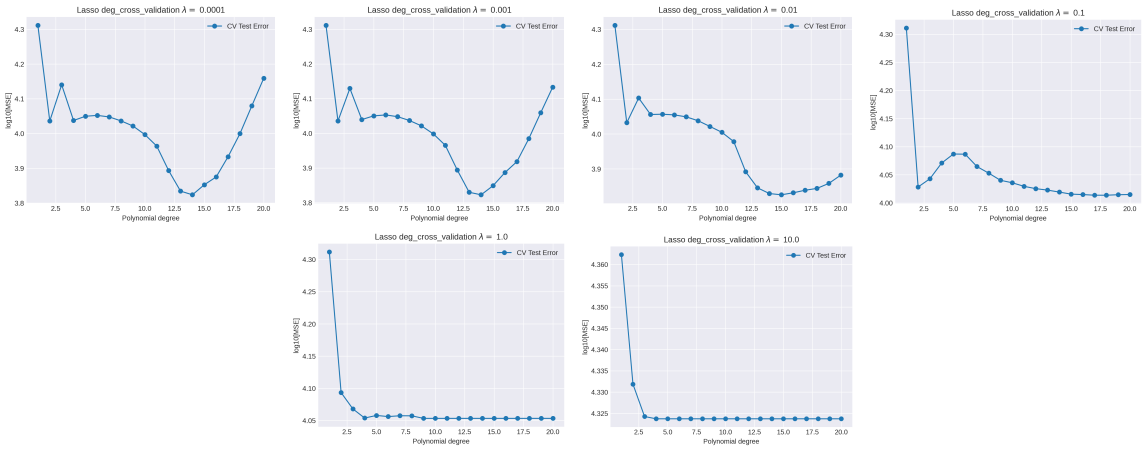

Figure 31. Terrain Lasso cross-validation up to the 20th polynomial degree for a range of λ.

REFERENCES

[1] G. James, D. Witten, T. Hastie and R. Tibshirani. *An introduction to Statistical Learning.* Springer, New York, 2017.

[2] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning.* Springer, New York, 2009.

[3] M. Hjort-Jensen. Applied Data Analysis and Machine Learning. Jupyter book, (2021)