FYS-STK4155 Project on Machine Learning
Part 1: Regression analysis and resampling methods

University of Oslo, Norway

Svein-Magnus Lommerud and Oliwia Malinowska

Fall 2022

Sammendrag

This is the first of three parts of a project on Machine Learning. Here, we have begun by introducing the different regression methods, like Ordinary Least Square, Ridge and Lasso regression methods. We have also studied the bias-variance trade-off in our model and implemented resampling techniques like bootstrap and cross-validation.

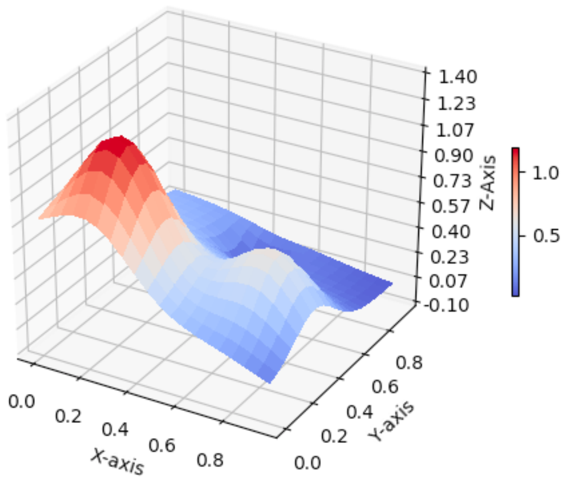
Introduction

Machine learning is a tool used to develop algorithms (like regression methods) that can recognize patterns in input data and use those patterns to make predictions about future data. In this first part of our project, we will look at three such algorithms, Ordinary Least Square (OLS), Ridge and Lasso regression methods. We will test their effectiveness by studying the Bias-Variance trade off and implementing resampling techniques such as bootstrap and cross-validation.
We will develop and verify those models based on the input data generated with the Franke's function and then, we will assess our models on real digital terrain data.

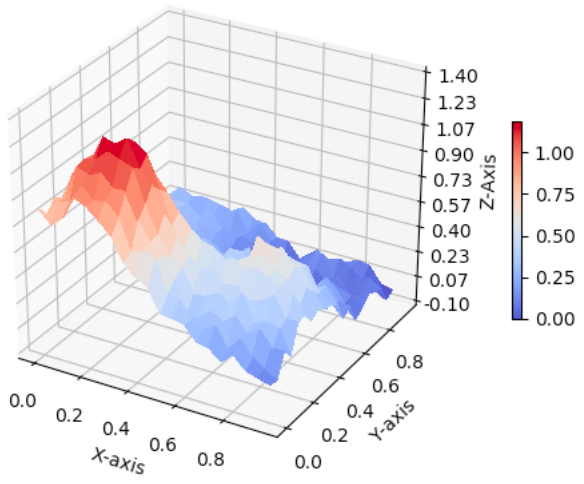
The Franke's function is given by following formula:

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

and will be defined for $x, y \in [0, 1]$. The three-dimensional plot of the Franke's function is given in Figure (1) without added noise, and with added noise in Figure (2).



Figur 1: Plot of the Franke's function without added noise.



Figur 2: Plot of the Franke's function with added noise.

Theory

Linear Regression

In linear regression method we want to find the best linear fit between the independent (y) and dependent variables (x):

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 ... + b_n x_n \tag{2}$$

and the optimal intercept b_0 and coefficient values $b_1, b_2, b_3, ..., b_n$ so that the error $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$ in 3 is minimized.[1]

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{3}$$

$f(\boldsymbol{x})$ is a continuous function that describes our data and we can approximate it with our model $\tilde{\boldsymbol{y}}$, where we minimize $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$ (Ordinary Least Squares Regression explained later), as follows:

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}, \tag{4}$$

where $\boldsymbol{X}$ is the design matrix with dimensions $n \times (p+1)$.

The expectation value of $\boldsymbol{y}$ for a given element i is given by:

$$\mathbb{E}(y_i) = \sum_j x_{ij}\beta_j = \boldsymbol{X}_{i,*}\boldsymbol{\beta}, \tag{5}$$

since:

$$\mathbb{E}(y_i) = \mathbb{E}[\sum_{j=0}^{p-1} x_{ij}\beta_j] + \mathbb{E}[\epsilon], \tag{6}$$

where $\mathbb{E}[\epsilon_i] = 0$ since ϵ is normally distributed.

The variance is given by:

$$Var(y_i) = \sigma^2, \tag{7}$$

since:

$$Var(y_i) = \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2, \tag{8}$$

and using $y_i = \boldsymbol{X}_{i*}\boldsymbol{\beta} + \epsilon_i$:

$$\begin{aligned}
Var(y_i) =& \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2 \\
&= \mathbb{E}[(\boldsymbol{X}_{i*}\boldsymbol{\beta} + \epsilon_i)^2] - \mathbb{E}[y_i]^2 \\
&= \mathbb{E}[\boldsymbol{X}_{i*\beta}^2 + \epsilon_i^2 + 2\boldsymbol{X}_{i*}\boldsymbol{\beta}\epsilon_i] - \mathbb{E}[y_i]^2 \\
&= \boldsymbol{X}_{i*}\beta^2 + \mathbb{E}[\epsilon_i^2] + 2\boldsymbol{X}_{i*}\boldsymbol{\beta}\mathbb{E}[\epsilon_i] - \mathbb{E}[y_i]^2 \\
&= \boldsymbol{X}_{i*}\beta^2 + \mathbb{E}[\epsilon_i^2] + 2\boldsymbol{X}_{i*}\boldsymbol{\beta}\mathbb{E}[\epsilon_i] - \mathbb{E}[\boldsymbol{X}_{i*}\beta + \epsilon_i]^2 \\
&= \boldsymbol{X}_{i*}\beta^2 + \mathbb{E}[\epsilon_i^2] + 2\boldsymbol{X}_{i*}\boldsymbol{\beta}\mathbb{E}[\epsilon_i] - \boldsymbol{X}_{i*}\boldsymbol{\beta}^2
\end{aligned}$$

and since $\mathbb{E}[\epsilon_i^2] = \sigma^2$ and $\mathbb{E}[\epsilon_i] = 0$, then:

$$Var(y_i) = \sigma^2 + 0 = \sigma^2. \tag{9}$$

Therefore, $y_i \sim N(\boldsymbol{X}_{i,*}\boldsymbol{\beta}, \sigma^2$, which means that $\boldsymbol{y}$ follows normal distribution with mean value $\boldsymbol{X\beta}$ and variance σ^2.[2]

Ordinary Least Square (OLS) method

One of the goals in this project, and the main idea of the model optimization is to find a set of β parameters that gives us the minimized cost-function defined as:

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})\}, \tag{10}$$

where n is the number of cases $i = 0, 1, 2, ..., n-1$, $\boldsymbol{y}$ is the set of response (outcome) variables y_i, and $\boldsymbol{X}$ is the design matrix of values of explanatory variables $\boldsymbol{x}_i = [x_{i0}, x_{i1}, ..., x_{ip-1}]$ (running from 1 to $p-1$).

The β parameters are therefore found by setting the derivative of the cost function to zero, resulting in the following way:

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \beta_j} = -\frac{2}{n}\left[\sum_{i=0}^{n-1} x_{ij}(y_i - \beta_0 x_{i,0} - \beta_1 x_{i,1} - \beta_2 x_{i,2} - ... - \beta_{n-1}x_{i,n-1}\right] = 0. \tag{11}$$

In the matrix-vector form it is given as:

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \beta} = 0 = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X\beta}), \tag{12}$$

and if the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, this will give us:

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{13}$$

where $\boldsymbol{X}^T\boldsymbol{T} \in \mathbb{R}^{p\times p}$.

When the goal of finding the right β parameters is reached, we can use them to fit the new values $\tilde{y} = \boldsymbol{X}\hat{\beta}$ (the hat means that they are optimal parameters).)

The expectation value $\mathbb{E}(\hat{\beta})$ is given by:

$$\begin{aligned}\mathbb{E}(\hat{\beta}) =& \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y})\end{aligned} \tag{14}$$

,

and since $\mathbb{E}(\boldsymbol{y}) = \boldsymbol{X\beta}$:

$$\mathbb{E}(\hat{\beta}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\beta = \beta. \tag{15}$$

Mean Squared Error (MSE)

To measure how accurate our model is in predicting the expected results, we can use mean squared error (MSE), which is probably the simplest type of loss function [3]. It is defined as:

$$MSE(\boldsymbol{y}, \boldsymbol{\tilde{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2, \tag{16}$$

where $\boldsymbol{\tilde{y}}_i$ is the predicted value (the output of our model) of the i-th sample, y_i is the corresponding true value (the expected results) and N is the number of samples we are testing against [3].

The advantage of MSE is that it will enlarge the error by squaring it so any data with large error in our trained model will easily be detected. At the same time, if there's only one bad prediction in our model, enhancing the error will not be as favourable [3].

R^2 score function

Another method used to evaluate the accuracy of a ML model is the R^2 score, also known as the coefficient of determination. It is defined as:

$$R^2(\boldsymbol{y}, \boldsymbol{\tilde{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2}, \tag{17}$$

where the mean value of $\boldsymbol{y}$ is defined as $\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$.

R^2 measures the amount of variance in the prediction values explained by the dataset [4], so if the R^2 score is close to 1, then it means that the variance is covered very well. Higher R^2 score means therefore that our model is better at predicting new data.

Bias-variance trade-off

The β parameters can be found by optimizing the mean squared error via the following cost function:

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}[(\boldsymbol{y} - \boldsymbol{\tilde{y}})^2], \tag{18}$$

where $\mathbb{E}$ is the sample value. This can be rewritten as:

$$\mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + \sigma^2, \tag{19}$$

with the calculation included in the Appendix A.

The first term is the square of the bias of the learning method. The second term is the variance and the last term is variance of the error ϵ.

The first two terms together (bias and variance) together give us the mean squared error.

Underfitting and overfitting

Real data usually consist of a large amount of data points and therefore you often use random subsets of it to train a ML model. Each subset can have different parameters and different amount of data. This means that the corresponding models (trained on the subsets of data) have some differences between each other. How similar those different models are to each other and how strong they are in making a prediction on new data, is the background for the bias-variance trade-off.
A ML model can either be improved by decreasing the variance, which is the difference between the predicted values coming from those different models, or by decreasing the bias, which is the average difference between the predicted values and the true values.
We can decrease the variance by simplifying the model to the point where all the different models (trained on different subsets of data) will give a similar prediction. It is in general a good thing, since we don't want the predictions to change a lot based on small differences in the training set. The disadvantage of an oversimplified model is that the bias is high, which means that on average, those different models give us quite inaccurate predictions. This low variance-high bias situation is called underfitting and means in general that our model is oversimplified and doesn't take into account all the parameters describing the data set, so although the predictions based on different subsets of data are consistent with each other, they are too far away from the true values.
As mentioned above, the other desired quality is a low bias, which means that, on average, predictions are close to the truth. A very low bias is achieved by including all the parameters describing the data set in a complex way. By doing that we end up with an overly complex model. The different subsets of data have each some noise that is specific for that single subset and is not representative of the whole data set. In case of an overly complex model, that is trained "too well", that noise is given too much weight. This situation is called overfitting. In this case, the differences between predictions coming from those different models are large, meaning the variance is high, but statistically, looking at the average of

those different models, we will get a good prediction.
The ideal scenario is a model with low bias and low variance, but to achieve both at the same time is impossible. By optimizing one of them, the other one gets increased. The aim is therefore to find an optimal trade-off, where the predictions are both close enough to the truth and consistent enough for the different training subsets of data.

Bootstrap

Bootstrapping is a powerful resampling technique that can be used to for example estimate the uncertainty of a model.

When analyzing data, even though we don't know the parameters of a population, we can recreate them from a single sample of population data with bootstrapping (if the sample is big enough, it resembles the population quite well already).

With bootstrapping, we resample the data from the sample *with replacement*. It means that we randomly select a data point from the sample and put it back before selecting again, allowing for a single datapoint to either be selected once, multiple times or never.

With the bootstrapped sample, we can calculate the mean value and repeat the process many times, until we get a sampling distribution of the mean values.

This technique is very useful for estimating parameters for data with unknown statistical properties or that lack a standard calculation (like coefficient of variation) [5].

A model that has been trained on a sample of data can be evaluated using the samples not included in the training. For each iteration of bootstrapping the model is fitted on a given sample of data and then is tested in predicting the data in the samples not included in the training [6].

Cross-validation

When we train our model, we look at the difference between the true data and the predicted data, and by that we evaluate our model in terms of how good it is based on the training data. To test our model's efficiency in terms of the *new data*, we use a process called cross-validation.
The simpler cross-validation method, called the holdout method, involves removing a part of the training data to use for the validation purposes. The removed part of the data is then used to make predictions based on the rest of the training data. This method is simple, but it involves high variance, because of the uncertainty of which data points will be removed and used in the validation process. If the splitting of data is random, there can be an uneven representation of the different samples in the training and testing shares of the data. By removing some of the

training data, we also increase the bias error, since some patterns can be lost in the process. To remove those obstacles, another cross-validation method is used, and it's called a k-fold cross-validation method.

In the k-fold cross-validation method, we structure the splitting of the data into k subsets. The holdout method mentioned above is then performed k times. For each iteration, on the k-subsets is removed and used as validation data, while the other subsets are used as training data. Which k-subset is used for validation changes for each iteration and we end up with an averaged error estimation. With this method, both the bias and variance is reduced, since we use most of the data for for both validation and training.[7]

Ridge and Lasso Regression

If our data set has few features and the test score is worse than the training score, it will lead to over-fitting. To prevent that and to reduce model complexity, we can use simple techniques like Ridge and Lasso Regression. [8]

Ridge regression is a linear regression model, which means that a linear function of input features is used to predict the output. To minimize the cost function, we need to optimize the input features.[8] The optimization problem is as follows [9]:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\left((\boldsymbol{y}-\boldsymbol{X}\beta)^T(\boldsymbol{y}-\boldsymbol{X}\beta)\right), \tag{20}$$

which we can modify using the definition of a norm-2 vector $\parallel \boldsymbol{x} \parallel_2 = \sqrt{\sum_i x_i^2}$ in the following way:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 = \frac{1}{n} \parallel \boldsymbol{y}-\boldsymbol{X}\beta \parallel_2^2 . \tag{21}$$

By adding the penalty term λ, which regularizes the β coefficients if they have large values [8], we get a new optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n} \parallel \boldsymbol{y}-\boldsymbol{X}\beta \parallel_2^2 + \lambda \parallel \beta \parallel_2^2, \tag{22}$$

and with the constraint on the coefficients: $\parallel \beta \parallel_2^2 \leq t$, where t is a finite number larger than zero, we end up up with the Ridge regression optimization problem. [9]

Lasso stands for least absolute shrinkage and selection operator and the Lasso regression optimization problem is given by:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n} \parallel \boldsymbol{y} - \boldsymbol{X}\beta \parallel_2^2 + \lambda \parallel \beta \parallel_1, \tag{23}$$

where we have used the definition of a norm-1 vector $\parallel \boldsymbol{x} \parallel_1 = \sum_i \mid x_i \mid$ [9], with the requirement of $\parallel \beta \parallel_1 \leq t$, where t is a finite number larger than zero.

Since in Lasso regression we don't take the square of the coefficients, we can end up with zero coefficients which means that some features can be selected out from the data set. This is in addition to preventing the over-fitting. [8]

Methods

Ordinary Least Square (OLS) on the Franke's function.

In this exercise we are generating our own data set for the two-dimensional Franke's function defined for $x, y \in [0, 1]$. We will perform the Ordinary Least Square (OLS) regression analysis (to minimize the cost function, find the β parameters and fit the Franke's function) and use polynomials in x and y up to fifth order to verify how effective this OLS model is in fitting our input data.
We will use both versions of Franke's function, with and without added noise.

Scaling the data

Before we can create a machine learning model we need to scale the data. Scaling of date is considered one of the key aspects of a good machine learning model. We will be using the Standard scaler. Standardization transforms the data to have zero mean and a variance of 1, they make our data unit-less. We need to scale the data if there is a large difference in the range of our data. If we for example have a data set that ranges from values in the hundreds to values in tens of thousands the larger numbers will fully dominate our prediction model.

Train-Test Split

For this exercise we will be utilizing the Train-Test Split method. The purpose of this method is to estimate how well the model preforms on new data. As inferred in the name of the method we split our data into two different sets, the training set and the testing set. In order for this to be usefull we need a large enough amount of data to begin with or else the sets will be to small and we will not be able to accurately mirror the original set of data. This will in turn lead us to be unable to evaluate the model. There are other methods that are more suitable for smaller sets of data like K-folding which we will be using in a later exercise. So assuming we have enough data we will need to decide how much of the data will be used for training and how much that will be used for testing. There is not any particular optimal ratio to split the data, but its common to use about 80% for training and 20% for testing.

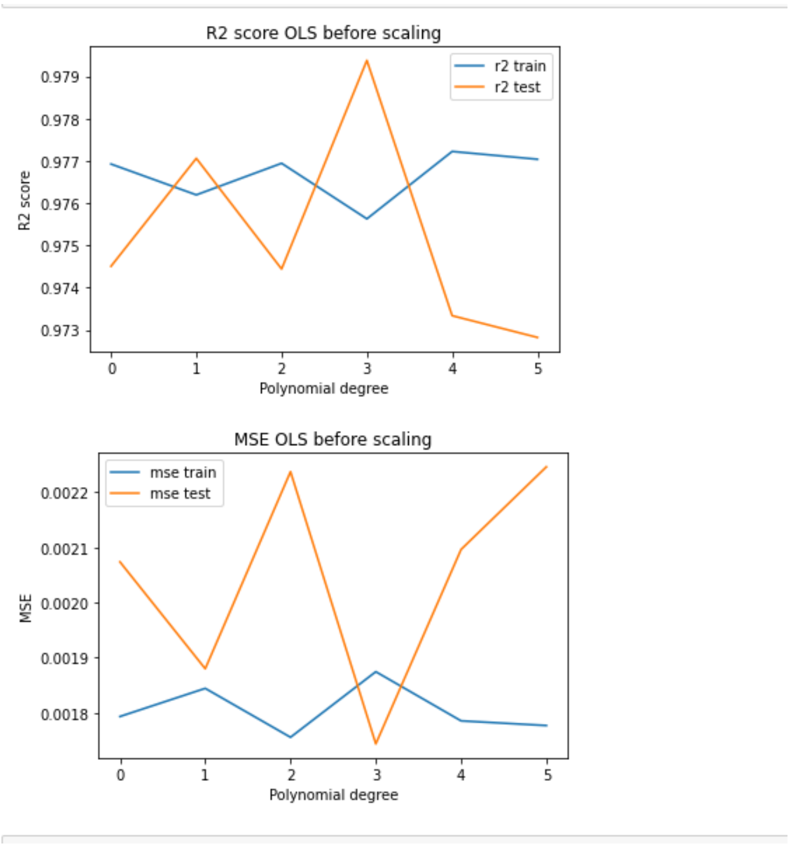
Bias-variance trade-off and resampling techniques

In this part we will study the bias-variance trade-off by implementing the bootstrap resampling technique.

Results

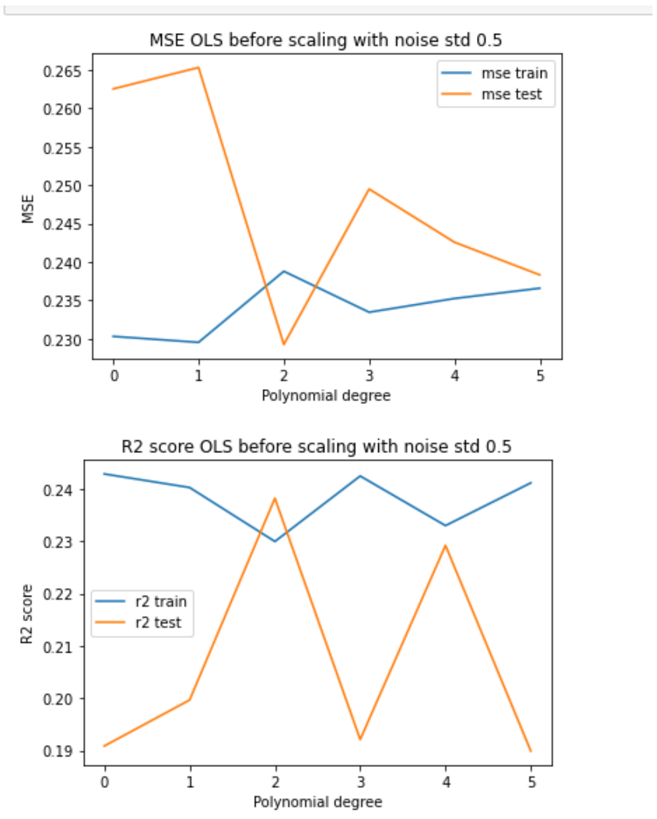
Ordinary Least Square (OLS) on the Franke's function

We have done four different runs. We have tested with and without scaling the data and with and without noise.



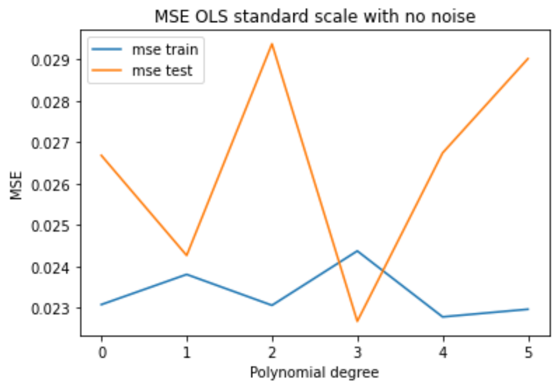
Figur 3: Plot of R2 and MSE scores with regards to polynomial degree before scaling of the data and no noise.

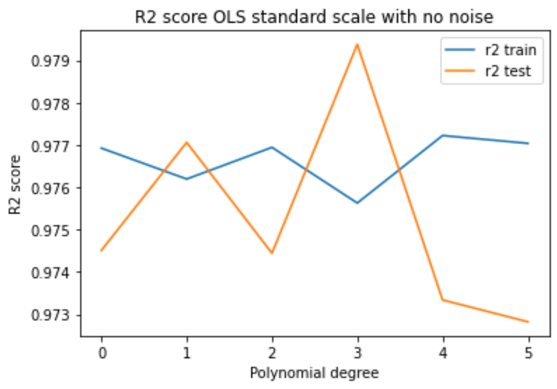
From Figure 3 we see that there are small differences for our R2 and MSE scores from polynomial degree 0 to 5. Although it may seem that both values are best at polynomial degree three. We have a very low MSE score and a R2 score close to 1 which are good indications that our model is good.



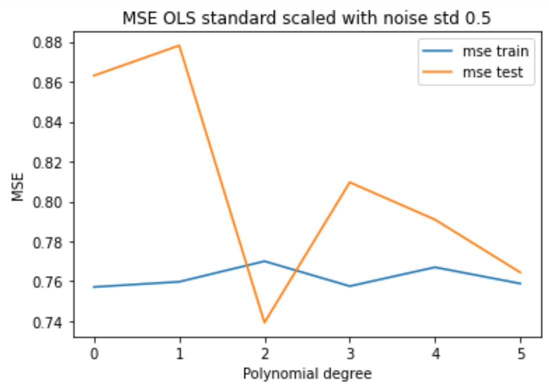
Figur 4: Plot of R2 and MSE scores with regards to polynomial degree before scaling the data and with noise.

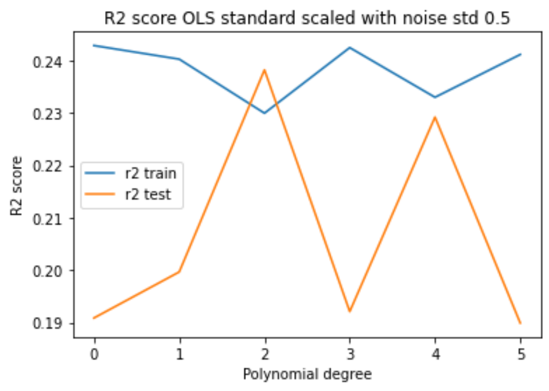
In figure 4 we see that our MSE score has increased significantly compared to Figure 3 and we also have a decline in our R2 score. As with our data from without noise we have the best results at polynomial degree three. Its apparent from the results that noise in the data play a important part in how well our model preforms.





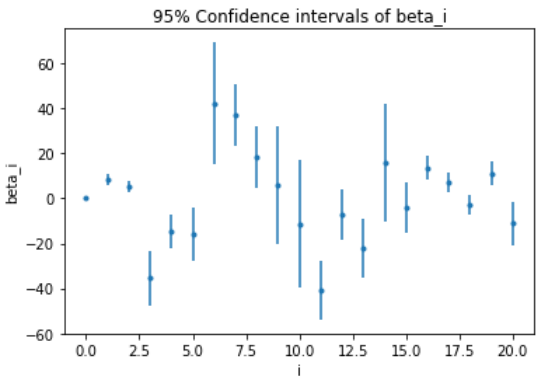
Figur 5: Plot of R2 and MSE scores with regards to polynomial degree after using the standard scaler and without noise.



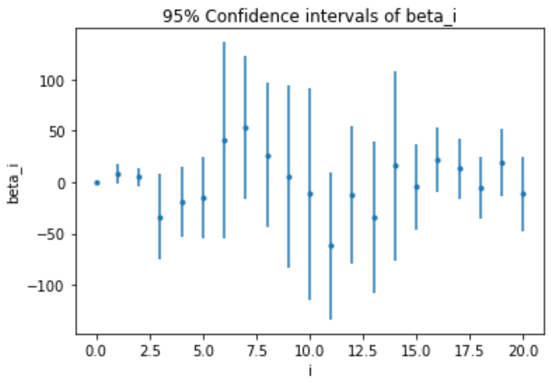


Figur 6: Plot of R2 and MSE scores with regards to polynomial degree after using the standard scaler and with noise.

Figure 5 and Figure 6 gives us the same results as the runs without scaling. Scaling is not always necessary. As for this particular case our data is not spanning a large enough range for scaling to have an affect, but if the data had more range in order of magnitude we would have lost outliers if we did not scale.



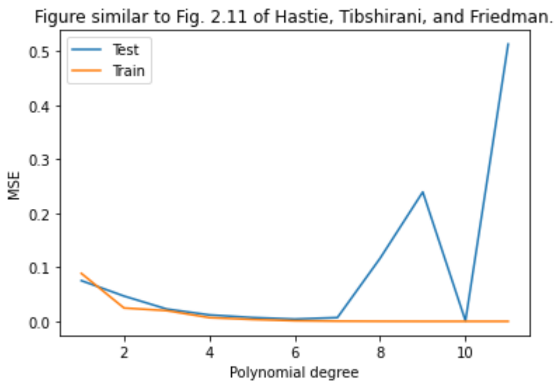
Figur 7: Plot of Beta confidence interval for data without noise



Figur 8: Plot of Beta confidence interval for data with noise

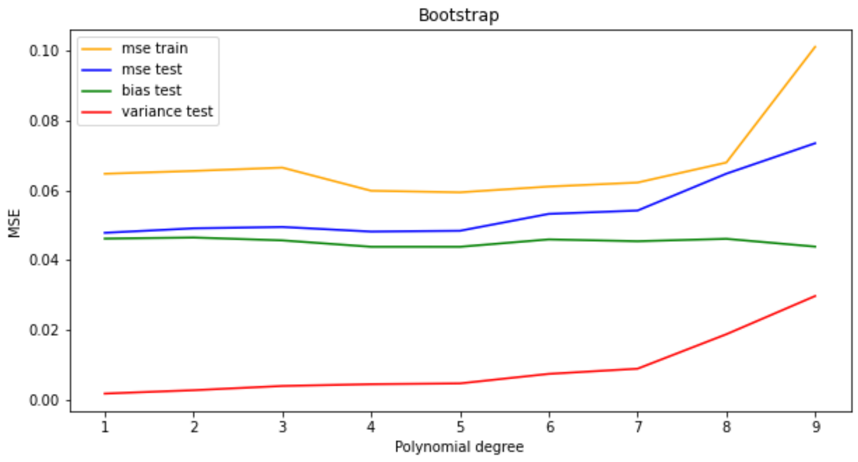
Bias-variance trade-off and resampling techniques

First we reproduced a figure simular to 2.11 of Hastie, Tibshirani, and Friedman.



Figur 9: Figure similar to Fig. 2.11 of Hastie, Tibshirani, and Friedman. Figure 2.11

Then we performed an analysis of the bias-variance trade-off on our test data. The trade-off between reducing the bias and variance of a model is important to understand so that we avoid underfitting or overfitting the data.



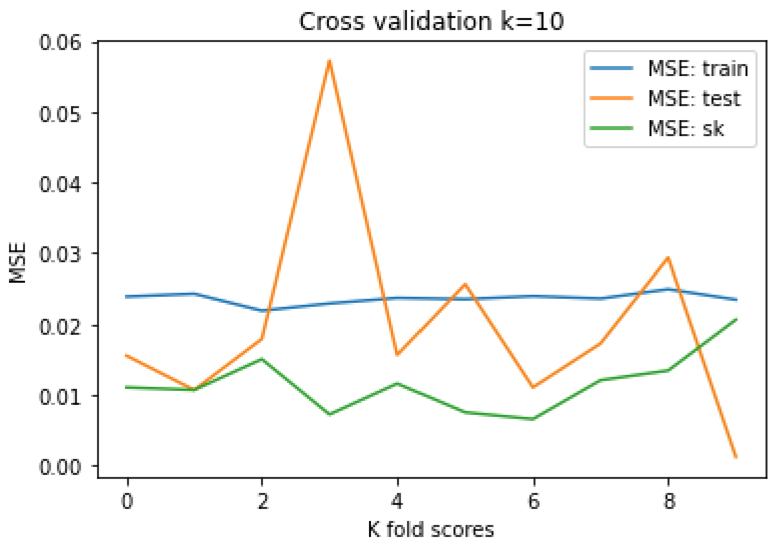
Figur 10: Plot of Bias-variance trade-off analysis using bootstrap

We plotted the Mean Squared Error for the test and train data, including the bias and variance part for the test data. From figure 10 we can deduct that for a higher polynomial degree the model becomes overfitted since the variance increases and the bias decreases. This part of the plot indicated that error rates on the test data is high, since our model is not adapting well to new data points that deviate from the rest, as mentioned above.

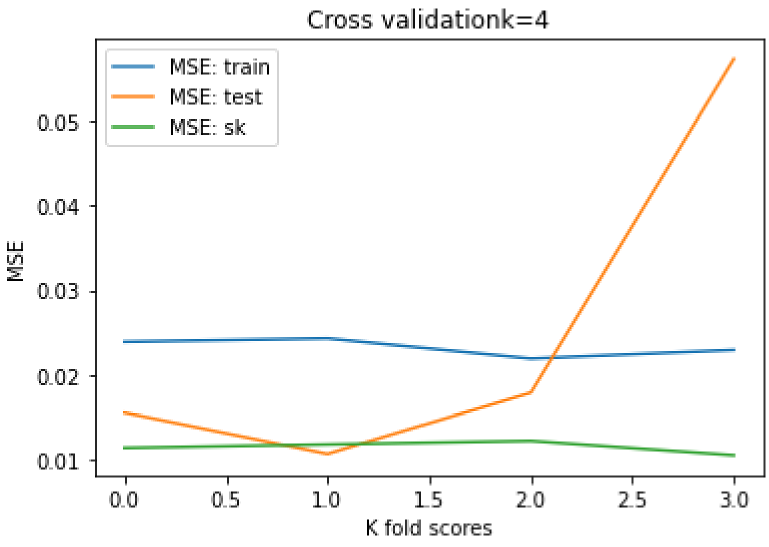
When our model passes degree 6 the Mean Squared Error for training data increases significantly compared to lower degrees. Unfortunately, we don't see any decrease in the MSE, but between polynomial degree 3 and 4, the bias line is slightly decreasing and in that interval we see the best possible bias-variance trade-off for this model.

Cross-validation as a resampling technique

Here we have used the cross-validation method to test the accuracy of a machine learning model on new (resampled) data and compared our code with that of scikit-learn.

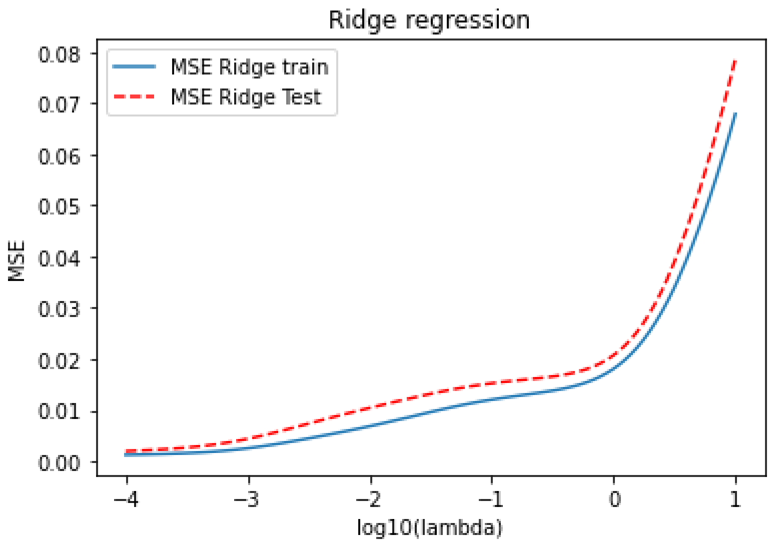


Figur 11: Plot of Cross validation



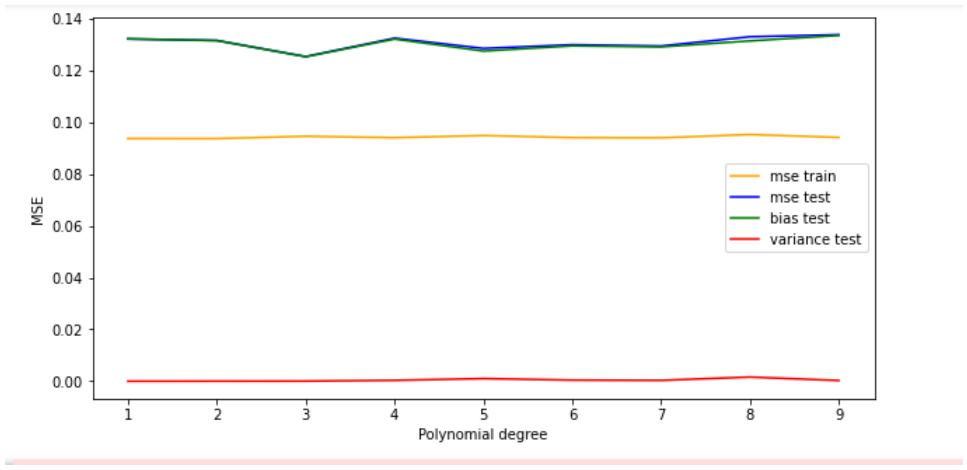
Figur 12: Plot of Cross validation

Ridge Regression on the Franke function with resampling



Figur 13: Plot of Ridge regression used on Franke's function

Lasso Regression on the Franke's function with resampling



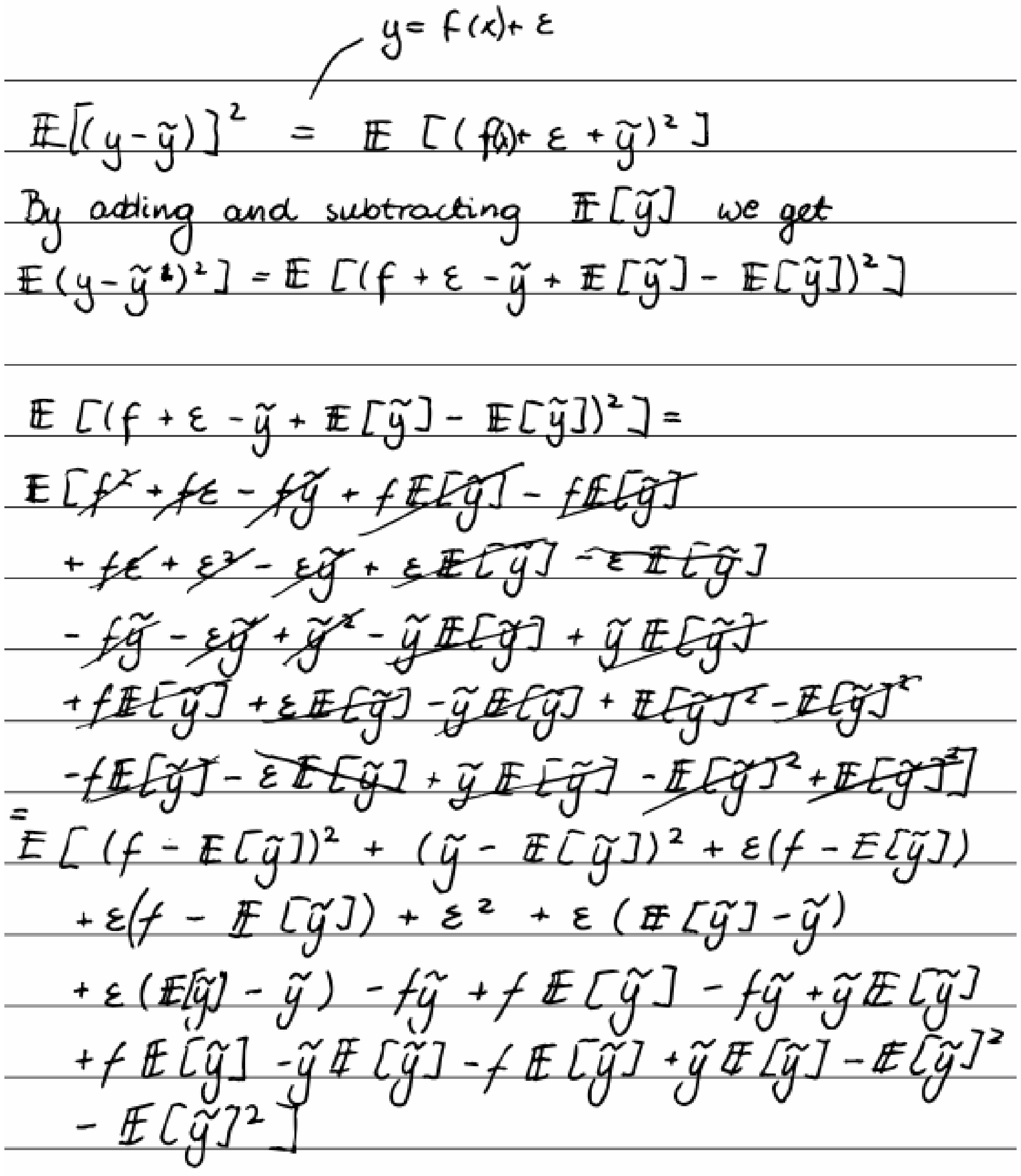
Figur 14: Plot of bias-variance trade-off for the Lasso Regression model used on Franke's function

Most of the mean squared error in the test data comes from the bias part.

Conclusion

In this part of the project we have achieved a much clearer understanding of the basics of Machine Learning. We have studied several regression methods (like OLS, Ridge and Lasso) and tested the efficiency of our model with resampling techniques (like bootstrap and cross-validation). We have tested the model with the help of R2 score and Mean Squared Error, with and without scaling the data, and with or without the added noise, with regards to the polynomial degree. We have performed the bias-variance trade-off analysis of our model. We could study our methods further by comparing them with the scikit-learn functionalities and attempt to use our models on new data sets.

A Rewriting the cost function for the bias-variance trade-off



$$y = f(x) + \varepsilon$$

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

By adding and subtracting $\mathbb{E}[\tilde{y}]$ we get

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

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

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

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

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

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

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

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

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

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

$$+ f\mathbb{E}[\tilde{y}] - \tilde{y}\mathbb{E}[\tilde{y}] - f\mathbb{E}[\tilde{y}] + \tilde{y}\mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}]^2$$

$$- \mathbb{E}[\tilde{y}]^2]$$

$$
\begin{aligned}
\mathbb{E}\Big[& \cancel{(f-\mathbb{E}[\tilde{y}])^2} + \cancel{(\tilde{y}-\mathbb{E}[\tilde{y}])^2} + \cancel{\varepsilon(f-\mathbb{E}[\tilde{y}])} \\
& + \cancel{\varepsilon(f-\mathbb{E}[\tilde{y}])} + \cancel{\varepsilon^2} + \cancel{\varepsilon(\mathbb{E}[\tilde{y}]-\tilde{y})} \\
& + \cancel{\varepsilon(\mathbb{E}[\tilde{y}]-\tilde{y})} - \cancel{f\tilde{y}} + \cancel{f\,\mathbb{E}[\tilde{y}]} - \cancel{f\tilde{y}} + \cancel{\tilde{y}\,\mathbb{E}[\tilde{y}]} \\
& + \cancel{f\,\mathbb{E}[\tilde{y}]} - \cancel{\tilde{y}\,\mathbb{E}[\tilde{y}]} - \cancel{f\,\mathbb{E}[\tilde{y}]} + \cancel{\tilde{y}\,\mathbb{E}[\tilde{y}]} - \cancel{\mathbb{E}[\tilde{y}]^2} \\
& - \cancel{\mathbb{E}[\tilde{y}]^2}\Big]
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}\Big[& (f-E[\tilde{y}])^2 + (\tilde{y}-\mathbb{E}[\tilde{y}])^2 + \varepsilon^2 \\
& + 2\varepsilon(f-\mathbb{E}[\tilde{y}]) + 2\varepsilon(\mathbb{E}[\tilde{y}]-\tilde{y}) \\
& - 2f\tilde{y} + \quad f\,\mathbb{E}[\tilde{y}] + \tilde{y}\,\mathbb{E}[\tilde{y}] - 2\,\mathbb{E}[\tilde{y}]^2\Big] \\
\mathbb{E}\Big[& (f-E[\tilde{y}])^2 + (\tilde{y}-\mathbb{E}[\tilde{y}])^2 + \varepsilon^2 \\
& + 2\varepsilon(f-E[\tilde{y}]) + 2\varepsilon(\mathbb{E}[\tilde{y}]-\tilde{y}) \\
& - (\mathbb{E}(\tilde{y})-\tilde{y})\cancel{\varepsilon}\cdot 2(f-E(\tilde{y}))\Big]
\end{aligned}
$$

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

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

$$+\underbrace{\mathbb{E}[(\mathbb{E}(\tilde{y})-\tilde{y})\cdot 2(f-\mathbb{E}[\tilde{y}])]}_{0}$$

0 (Expectation value of an expectation value is the expectation value)

Using $\mathbb{E}(\varepsilon)=0$ (ε is normally distributed)

$E(\varepsilon^2)=\sigma^2$

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

$$\mathbb{E}[(y-\tilde{y})^2] = \frac{1}{n}\sum_i (f_i-\mathbb{E}[\tilde{y}])^2 + \frac{1}{n}\sum_i (\tilde{y}_i-\mathbb{E}[\tilde{y}])^2 + \sigma^2$$

Referanser

[1] Deepanshi, «All you need to know about your first Machine Learning model – Linear Regression,» mai 2021.

[2] M. Hjorth-Jensen, «Project 1 on Machine Learning, deadline October 7, 2022,» sep. 2022.

[3] G. Seif, «Understanding the 3 most common loss functions for Machine Learning Regression,» mai 2019.

[4] A. Kharwal, «R2 Score in Machine Learning,» jun. 2021.

[5] M. E. Clapham, «26: Resampling methods (bootstrapping),» feb. 2016.

[6] J. Brownlee, «A Gentle Introduction to the Bootstrap Method,» mai 2018.

[7] P. Gupta, «Cross-Validation in Machine Learning,» jun. 2017.

[8] S. Bhattacharyya, «Ridge and Lasso Regression: L1 and L2 Regularization,» sep. 2018.

[9] M. Hjorth-Jensen, «Applied Data Analysis and Machine Learning,» 2021.