Bias Variance Trade-off

Analyzing Terrain Data

Bendik Nyheim, Marcus Moen, & Mira Mors

https://github.com/benyhh/FYS-STK4155

Department of Physics
University of Oslo, Norway
December 2021

Abstract

We have done a Bias-Variance tradeoff analysis for the linear regression models Ordinary Least Squares (OLS), Ridge and Lasso, a deep learning model using Feed Forward Neural Networks, and the ensemble method Decision Tree. The models were built based on a dataset of a terrain in Norway. We found that the bias and variance behaves differently for all of these models when the complexity varies. OLS had high bias and low variance for low complexities and low bias and high variance for high complexities. Lasso and Ridge behaved similarly as OLS, but the regularization term reduces this effect significantly. Decision trees had high variance and bias for low complexity, and low variance and bias for high complexity. Neural Network had high variance and bias for low complexities, and high variance and low bias for high complexities.

Contents

1 Introduction

The bias-variance tradeoff describes the problem of simultaneously minimizing two sources of error: bias and variance. It complicates the generalization of training data by supervised learning algorithms. The bias-variance decomposition provides a way to analyze the expected generalization error of a learning algorithm with respect to a given problem. It can be represented as a sum of three terms: the bias, the variance and and irreducible error which results from the noise of the problem itself. The bias-variance dilemma applies to all forms of supervised learning.

Here, we will look at regression using linear regression, feed forward neural network and decision trees. We do this by analysing how the bias and variance behaves when the complexity of a given model increases. Which methods we use, and how we define the complexity of a given model is presented in the methods section. The mathematical details behind the bias-variance decomposition are also explained there, and how estimates are obtained using the bootstrap resampling technique. The results are presented in the results section, and the analysis is found in the discussion section. At the end, everything is wrapped up in the conclusion.

2 Methods

Most of the methods used in this task have already been implemented in other works of ours. A short summary and/or references are given.

2.1 Two-dimensional Polynomial

The following definitions are taken from [2]. We define a two dimensional polynomial of degree d to be

$$f(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{i} \beta_{i-j,j} x^{i-j} y^j \tag{1}$$

where β are coefficients.
In matrix notation we have

$$f(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta} \tag{2}$$

where $\mathbf{x} = [1, x, y, x^2, xy, y^2, \cdots, x^d, x^{d-1}, \cdots, y^d]$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_p]^T$.

Thus, a two dimensional polynomial of degree d contains p terms, where

$$p = \frac{(d+1)(d+2)}{2} \tag{3}$$

2.2 Linear Regression

The following definitions are taken from [2]. The equation for the approximated $\tilde{y}$-values is

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

where $\boldsymbol{X}$ is the design matrix. $\boldsymbol{\beta}$ are the parameters to be determined.

Ordinary Least squares The cost for linear regression is given by

$$C(\boldsymbol{\beta}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{5}$$

where $\mathbf{y}$ are the target values and $\mathbf{X}$ is the input data (design matrix). $\boldsymbol{\beta}$ is a vector of weights, where each parameter is associated with a particular feature in the design matrix. In view of equation (5), the derivative of the cost function is defined as

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right) \tag{6}$$

it can be shown that the optimal parameters $\hat{\beta}$ is

$$\hat{\boldsymbol{\beta}}_{OLS} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{7}$$

Ridge Regression The cost for ridge regression is given by

$$C(\boldsymbol{\beta}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{2}\lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{8}$$

The optimal parameter can be shown to be

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{9}$$

Lasso For Lasso, there is no direct analytic expression for the optimal parameter. The cost function is given by

$$C(\boldsymbol{\beta}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda|\boldsymbol{\beta}| \tag{10}$$

Complexity For these regression models, we define the complexity as the two dimensional polynomial degree presented in section 2.1, not the number of features the design matrix has. So a linear regression model with complexity 10 has $11 \cdot 12/2 = 66$ columns (features) in the design matrix.

2.3 Feed Forward Neural Network

We have explained feed forward neural network for regression in [2]. Importantly, a liner activation function $f(x) = x$ is used in the last layer such that the output ($a_k = z_k$) in the last layer is the predicted value. For the hidden layers, we will use the ReLU activation function. The other parameters chosen are Adam as solver, regularization parameter $\lambda = 0.0001$, learning rate $\eta = 0.1$, batch size 200, momentum $\gamma = 0.9$, and two hidden layers. Here, we define the complexity of the model as the number of nodes for a fixed number of hidden layers. The explanation of these hyperparameters can be found in [2], except for the solver Adam, which is presented in [4].

2.4 Decision Tree

As in another of our projects [4] we will use decision trees, but now for regression and not classification. The difference to classification is that a regression tree is used to predict a quantitative response rather than a qualitative one. First, the predictor space, which is made up of the set of possible values $x_1, x_2, \ldots, x_p$, is split into J distinct and non-overlapping regions $R_1 \ldots R_J$. Every observation assigned to a region R_j is ascribed the same prediction. The prediction is simply the mean of the response values for the training observations in R_j. The regions are defined as high-dimensional rectangles or boxes. The rectangles are chosen such that the MSE becomes minimized

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \overline{y}_{R_j})^2, \tag{11}$$

where $\overline{y}_{R_j}$ is the mean response for the training observations within box j. As it is infeasible to examine all possible partitions of the feature space into J boxes, a top-down approach is implemented.

First, at the top of the tree, all observations are assigned to one region. Then, the predictor space is successively split. Each split consists of two new branches. The binary splitting into two new regions R_1 and R_2 is conducted by selecting the predictor x_j and a cutpoint s

$$\{X|x_j < s\} \quad \text{and} \quad \{X|x_j \geq s\}, \tag{12}$$

Then according to equation (11), the MSE, which is the sum of the MSE for each respective region, is minimized. The optimal pair of j and s can be found quickly if the number of features p is not too large. Moving down the tree, the process is repeated; looking for the best predictor and best cutpoint. Only the two previous identified regions are considered. A region will be split long as the MSE is decreased, unless there is only one sample left in the region. The complexity of the model is defined as the depth of the tree, which can by counted by the maximum number of decisions it can make before coming to a prediction. [1]

2.5 Bias-Variance Tradeoff

The principles of bias-variance tradeoff have been explained in [3]. Considering a dataset $\mathcal{L}$ consisting of the data $\mathbf{X}_\mathcal{L} = \{(y_j, \boldsymbol{x}_j), j = 0 \dots n-1\}$, it is assumed that the true data is generated from a noisy model

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{13}$$

where ϵ is normally distributed with mean zero and standard deviation σ^2. Predictions are done according to equation (4).
The bias is defined as the difference between the expectation value of the prediction , $\mathbb{E}[\tilde{y}]$ and the true value $f(x)$

$$Bias = \mathbb{E}[\tilde{y}] - f(x) \tag{14}$$

The variance is defined as the mean squared deviation of $\tilde{y}(x)$ from its expected value $\mathbb{E}[\tilde{y}(x)]$

$$Var(\tilde{y}(x)) = \mathbb{E}[\tilde{y}^2] - (\mathbb{E}[\tilde{y}])^2 = \mathbb{E}\left[(\mathbb{E}[\tilde{y}] - \tilde{y})^2\right] \tag{15}$$

The mean squared error can be written as a sum of the bias, variance and irreducible error

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \underbrace{\frac{1}{n}\sum_i \left[(\mathbb{E}[\tilde{\boldsymbol{y}}] - f(x_i))^2\right]}_{\text{average bias squared}} + \underbrace{\frac{1}{n}\sum_i \left[(\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\right]}_{\text{variance}} + \underbrace{\sigma_\epsilon^2}_{\text{irreducible error}} \tag{16}$$

where $\mathbb{E}_n$ denotes the expectation over the distribution of different test points x.

2.6 Evaluation

Each model is evaluated using the mean squared error (MSE)

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

where $\tilde{y}_i$ denotes the predicted target and y_i the true target value.

Bootstrap is the resampling technique we use to estimate the bias and variance. The idea of bootstrapping is drawing samples from the training data while the test data remains unchanged. The drawing is done with replacement, which means the same sample can be drawn multiple times in a given bootstrap cycle. This data is used to make a model, which then again is used to make a prediction. This is repeated a number of times, and the bias and variance can be calculated from all the bootstrap samples.

3 Data

For the bias-variance tradeoff analysis, we used terrain data from Norway obtained from the website Earth Explorer. The methods chosen in our analysis can be computationally heavy, so we only considered a small subset of a 50×50 grid that equals 2500 datapoints. The data contains the altitude of the terrain, and we made our own x- and y-axes ranging from 0 to 49. Only using this, the results of a linear regression model would be a flat plane. In order to increase the complexity of the linear regression models, we use two dimensional polynomials, presented in section 2.1, with our x- and y-values. The data is split into training and test sets, with a ratio of 80% and 20% respectively. The data is scaled by centering and dividing by the standard deviation. The surface the models are trying to fit is given in figure 1.

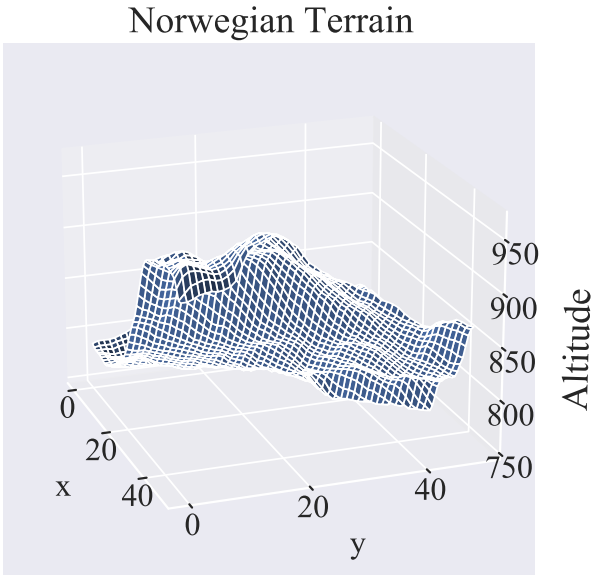


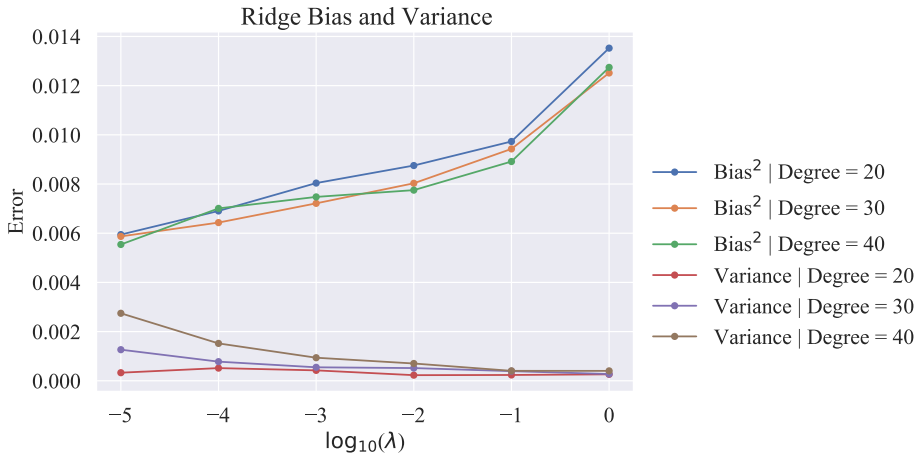
Figure 1: Surface of terrain in Norway.

4 Results

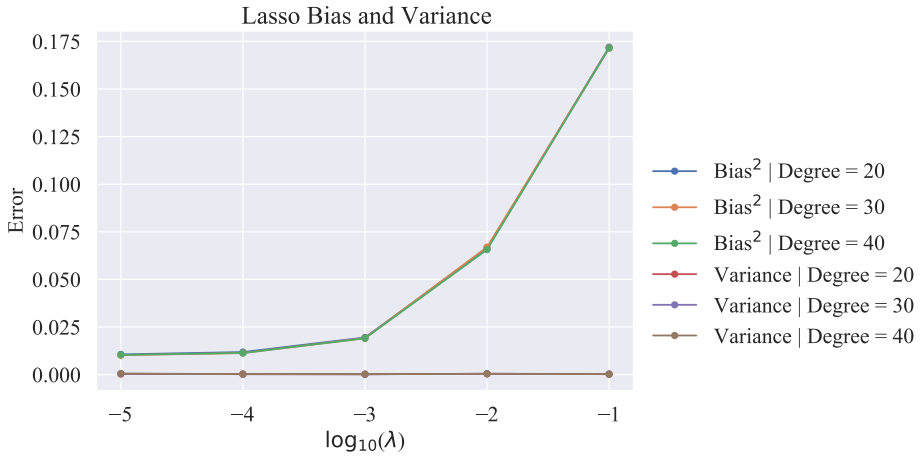
Figure 3 shows the bias-variance tradeoff for the the five models: OLS (figure 3a), Ridge (figure 3b) Lasso (figure 3c), a feed forward neural network with an architecture given in section 2.3 (figure 3d), and a decision tree (figure 3e). For Ridge and Lasso, we used the regularization parameter $\lambda = 5 \cdot 10^{-4}$ and $\lambda = 5 \cdot 10^{-6}$, respectively.

A bias-variance tradeoff as a function of the regularization term in Ridge and Lasso is shown in figure 2. The respective number of bootstrap cycles is specified in the figure text.

Table 1 shows the number of leaves for a given depth of the decision tree in one of the bootstrap cycles.

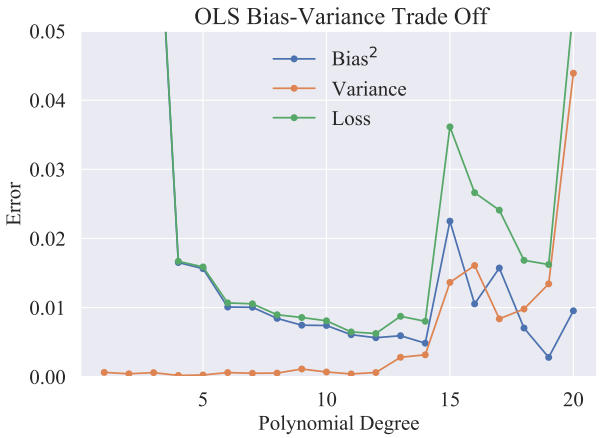


(a) Ridge, 50 bootstrap cycles

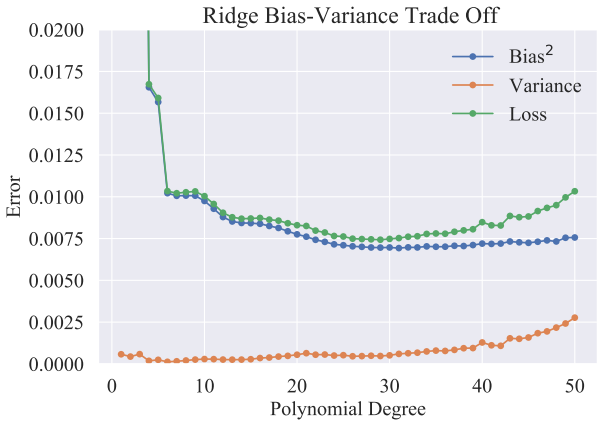


(b) Lasso, 50 bootstrap cycles

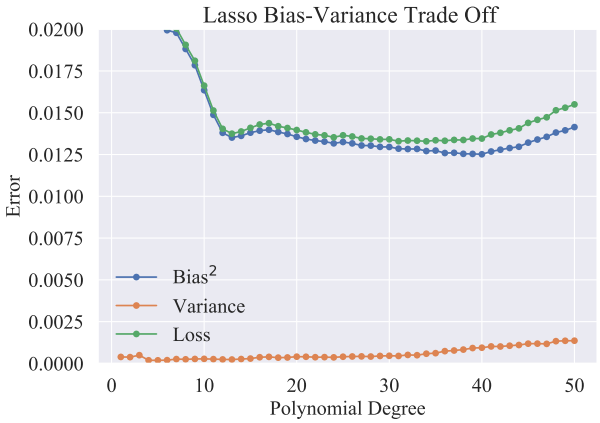
Figure 2: Bias and Variance for various complexities (specified by the degree) of Ridge and Lasso regression as a function of the regularization parameter λ.



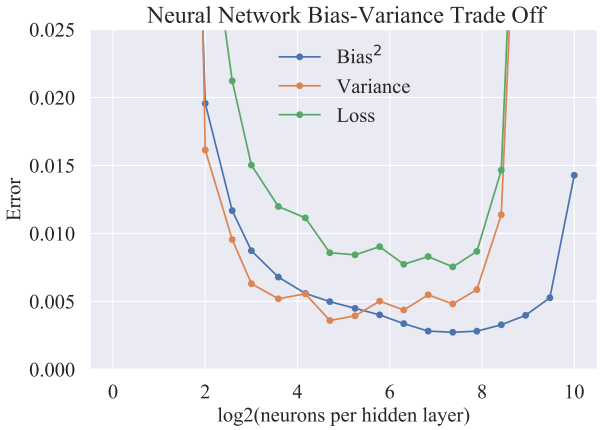
(a) Ordinary Least Square, 500 bootstrap cycles



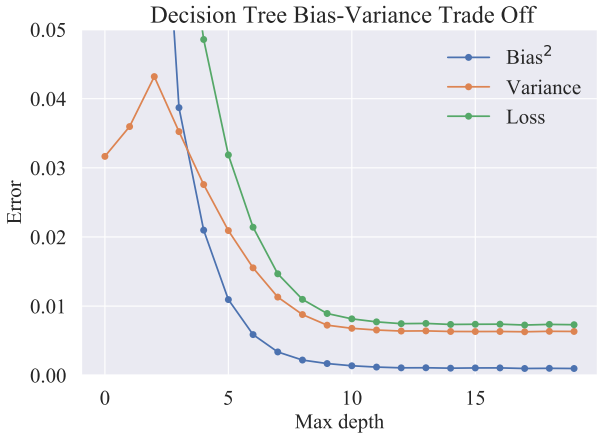
(b) Ridge, regularization parameter $\lambda = 5 \cdot 10^{-4}$, 500 bootstrap cycles



(c) Lasso, regularization parameter $\lambda = 5 \cdot 10^{-6}$, 200 bootstrap cycles



(d) Feed Forward Neural Network, 100 bootstrap cycles



(e) Decision Tree, 300 bootstrap cycles

Figure 3: Bias-Variance Analysis for Ordinary least square, Ridge, Lasso, and Feed Forward Neural Network. For the linear regression methods, the polynomial degree is the complexity of the model. For the Neural Network, the number of nodes is each of the two layers is the complexity, shown on a logarithmic scale with base 2 in the plot. The depth is the complexity of the decision tree.

Table 1: The number of leaves for a given depth of the decision tree for one of the bootstrap cycles.

Depth	1	4	8	12	16	17	18	19	20
Leaves	2	16	255	1239	1658	1703	1731	1757	1781

5 Discussion

In general, we expect the loss and variance to increase and the bias to decrease when the model complexity gets too high. This is because an arbitrarily high complexity will be able to fit all the training data perfectly. We then, however, expect the model to generalize poorly. It will perform worse on new data, and thus the variance gets high. In figure 3a we see that a 2D-polynomial degree of 12 has the lowest expected loss. When increasing the polynomial degree further, the variance drastically increases while the bias decreases. In the figure, we see the bias jumping a bit up and down when the complexity increases, and to be honest we are not quite sure as to why. Adding more complexity shouldn't increase the bias, so this should be investigated further. 500 bootstrap cycles are used, so the irregularity is likely not because of randomness.

The regularization parameter in Ridge and Lasso regression is introduced to prevent this overfitting by putting constraints on the regression coefficients. We can see this in figure 3b and 3c as the models have a much higher polynomial degree, while still not overfitting. So for the bias-variance tradeoff with Ridge and Lasso, it is also important to consider how the bias and variance behaves when the regularization parameter varies. This is shows in figure 2. As expected, the variance decreases when the regularization parameter increases, and this is because the regularization parameter puts constraints on the regression coefficients. We also see that while this happens, the bias increases, as the models doesn't fit as well to the training data. This is very apparent for Ridge 2a, but not as apparent for Lasso 3c. This is mostly because for this particular dataset, a very low value of the regularization parameter λ is needed. Additionally, it is very computationally heavy to compute for high complexities.

Now on to the neural network shown in figure 3d, which has quite a different tradeoff than that of the regression methods. The x-axis has a logarithmic scale with base 2, so the complexity of the neural network is actually increasing quite significantly. One of the strengths of neural networks is that they can fit very nicely to any type of data, but that comes with difficult interpretability. With low complexity, we see that both the variance and bias are high. For a linear regression model with low complexity, the variance is low because all the models are quite similar despite the different bootstrap samples. For a neural network however, the weights of the neurons can be quite different for each bootstrap cycle, and this might be the reason for a high variance with low complexity. When the complexity increases, both the variance and bias decreases. Why the variance is high for both low and high complexities, and low in the middle, is a little hard to understand, which again takes us back to hard interpretability of neural network. A way to understand this better would be to actually plot the predicted surface for different complexities, and see how it varies. The reason why we see the bias increasing with the complexity is likely due to the fact that the weights of the neurons are not converged. Increasing the number of epochs would likely yield lower bias, but since it took a long time to compute, we'll leave it as is.

Decision trees is definitely the method with the most surprising results (figure 3e). The x-axis shows the value of the maximum depth parameter, but even though the maximum depth increases, the number of leaves remains rather similar (table 1). The reason for this is that most of the training samples are fit perfectly. For some machine learning methods, this would lead to extremely high variance. However, the results on test data are clearly reasonable, which can be explained by the structure of the terrain data and how decision trees work. A region fits a training sample perfectly

and a test sample landing in the same region will likely have quite similar altitude, as the terrain doesn't typically vary drastically up and down. This is different from for instance linear regression models, where a perfect fit on all training data would lead to extremely high variance, and that is because the value between the interpolated training samples can be quite different, while a region in the decision tree will have constant value over a domain.

6 Conclusion

In the Bias-Variance tradeoff analysis for three different type of machine learning methods, we have found that the tradeoff is quite different for all of them. We saw that for linear regression models, the variance is low and bias is high for low complexities, while they switch when the model complexity increases. A regularization term can keep the models from overfitting. In a Feed Forward Neural Network however, both the bias and variance are high for low complexities, and while the bias keeps decreasing when the complexity is increased, the variance has a bottom before blowing up again. For the Decision Tree, both the Bias and Variance starts off high, and they both decrease with model complexity. The bias goes towards zero as the training samples are fit perfectly, while the variance stabilizes.

Further analysis could include investigating why the variance is high for low complexities of Neural Network and Decision Tree. A way to do this is plotting multiple predicted surfaces for different complexities of these models, which may give a lot of insight. Analysis of how the regularization affects the bias and variance for a neural network could also be done. It would also be interesting to see if the Bias-Variance tradeoff for these models are similar for a very different type of dataset.

References

[1] Morten Hjorth-Jensen. *Applied Data Analysis and Machine Learning: Decision trees*. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter6.html. Nov. 2021. (accessed: 14.12.2021).

[2] Bendik Nyheim, Marcus Moen, and Mira Mors. *Classification and Regression: From Linear and Logisitc Regression to Neural Networks*. Nov. 2021. URL: https://github.com/benyhh/FYS-STK4155/tree/main/Project2/Report. (accessed: 10.12.2021).

[3] Bendik Nyheim, Marcus Moen, and Mira Mors. *Regression Analysis and Resampling Methods, Using Ordinary Least Squares, Ridge Regression and Lasso Regression on the Franke Function and Real Terrain Data*. https://github.com/benyhh/FYS-STK4155/tree/main/Project1/Report. Sept. 2021. (accessed: 14.12.2021).

[4] Bendik Nyheim, Marcus Moen, and Mira Mors. *Water Potability Prediction: Binary Classification using Deep Learning and Ensemble Methods*. Nov. 2021. URL: https://github.com/benyhh/FYS-STK4155/tree/main/Project3/Report. (accessed: 10.12.2021).