University of Oslo

FYS-STK3155

Project 3

Brage Wiseth bragewi@ifi.uio.no
Felix Cameren Heyerdahl felixch@ifi.uio.no
Eirik Bjørnson Jahr eiribja@ifi.uio.no

December 9, 2023

https://github.com/bragewiseth/MachineLearningProjects

Contents

Abstract

Lorem ipsum dolor sit amet, officia excepteur ex fugiat reprehenderit enim labore culpa sint ad nisi Lorem pariatur mollit ex esse exercitation amet. Nisi anim cupidatat excepteur officia. Reprehenderit nostrud nostrud ipsum Lorem est aliquip amet voluptate voluptate dolor minim nulla est proident. Nostrud officia pariatur ut officia. Sit irure elit esse ea nulla sunt ex occaecat reprehenderit commodo officia dolor Lorem duis laboris cupidatat officia voluptate. Culpa proident adipisicing id nulla nisi laboris ex in Lorem sunt duis officia eiusmod. Aliqua reprehenderit commodo ex non excepteur duis sunt velit enim. Voluptate laboris sint cupidatat ullamco ut ea consectetur et est culpa et culpa duis.

Keywords: Regression, Classification, Neural Networks

Introduction

Wave equation is a partial differential equation that describes the propagation of waves through a medium. blah blah blah
The structure of this report is as follows:

- In Section 2, we introduce the logistic regression model.
- In Section 3, we discuss gradient descent and backpropagation, key techniques in the optimization of machine learning models.
- In Section 4, we delve into the fundamentals of neural networks.
- Section 5 covers the data used in our experiments and analyses.
- Section 6 presents our results and provides a discussion on their implications.
- Section 7 concludes the report, summarizing key findings and insights.
- Appendix A explores regression using neural networks.
- Appendix B investigates the universal approximation theorem and its applications.

1. Wave Equation

The wave equation is a partial differential equation that describes the propagation of waves through a medium. It is given by

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{1}$$

where $u(x,t)$ is the displacement of the wave at position x and time t, and c is the wave speed. This equation can be solved analytically for a given initial condition, $u(x,0)$, and initial velocity, $\frac{\partial u}{\partial t}(x,0)$, using the method of separation of variables. However, this method is not always feasible, particularly when the initial conditions are complex. In such cases, numerical methods

2. CNN

Convolutional neural networks (CNNs) are a type of neural network that are particularly effective in image classification tasks. They are comprised of convolutional layers, which apply a convolutional filter to the input, and pooling layers, which downsample the input. The convolutional filters are typically small, such as 3×3 or 5×5, and are applied to the input image in a sliding window fashion. The filter is multiplied element-wise with the input image, and the resulting values are summed to produce a single value. This process is repeated for each position in the input image, resulting in a new image with the same dimensions as the input. The pooling layers downsample the input by applying a function to a small region of the input, such as the maximum or average value. This process is also repeated for each position in the input, resulting in a new image with smaller dimensions. The convolutional and pooling layers are typically interleaved, with the pooling layers serving to reduce the dimensionality of the input between convolutional layers. The final layer of the network is typically a fully connected layer, which is a standard neural network layer where each neuron is connected to every neuron in the previous layer. The output of this layer is then used to classify the input image. The convolutional layers are able to extract features from the input image, while the fully connected layers are able to classify the image based on these features. This structure allows CNNs to achieve high accuracy in image classification tasks. In this project, we will explore the application of CNNs to the wave equation.

3. RNN

Due to the sequential nature of RNNs, they might prove useful in solving time series problems such as the wave equation. In this section, we will explore the application of RNNs to this problem.

4. Data

For our classification task, we will utilize the widely recognized Wisconsin Breast Cancer Dataset[10]. This dataset comprises 569 data points, each with 30 distinctive features. These features are derived from digitized images of a fine needle aspirate (FNA) of breast masses, focusing on various characteristics of the cell nuclei depicted in the images. The dataset quantifies several attributes for each cell nucleus, including radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. For each attribute, three types of measurements are provided: the mean, standard error, and the "worst" or largest (which represents the mean of the three largest values). This results in a total of 30 features for each data point. For instance, field 1 represents the Mean Radius, field 11 is Radius SE, and field 21 corresponds to the Worst Radius. All feature values in this dataset are recorded with four significant digits, and there are no missing attribute values. The class distribution within the dataset is as follows: 357 benign cases and 212 malignant cases. The primary objective is to classify these tumors as either benign or malignant based on the features provided. In this context, a positive result indicates a benign tumor, while a negative result signifies a malignant tumor. Essentially, our goal is to accurately identify cases of cancer and classify them as negative.

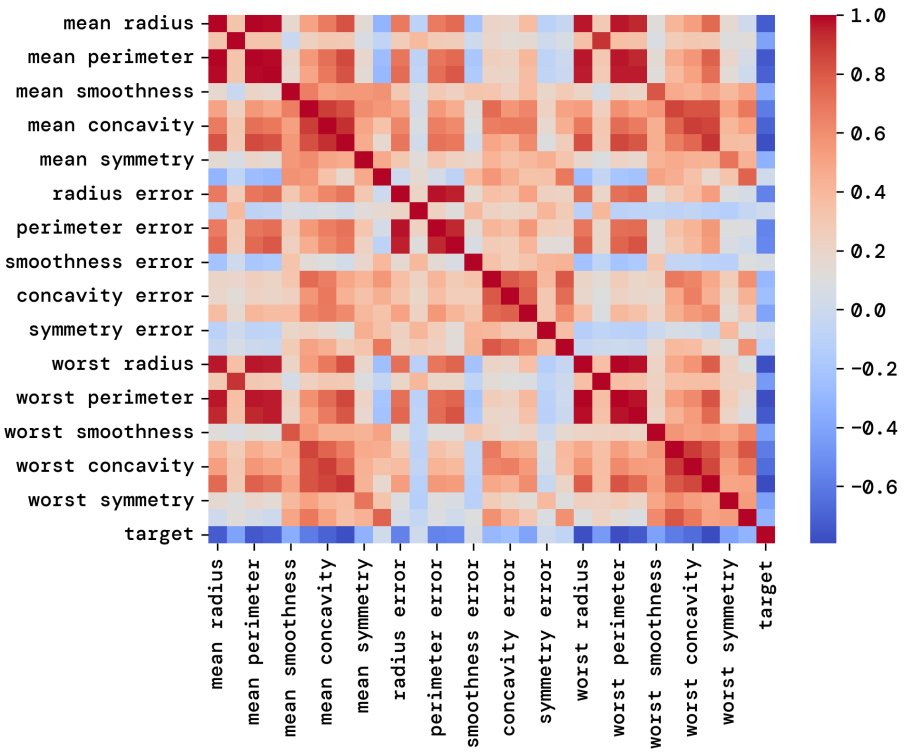

Figure 1: Feature correlation matrix. This provides insight into the redundancy among features. Only every other feature is annotated, but the missing labels can be inferred from Figure 2.

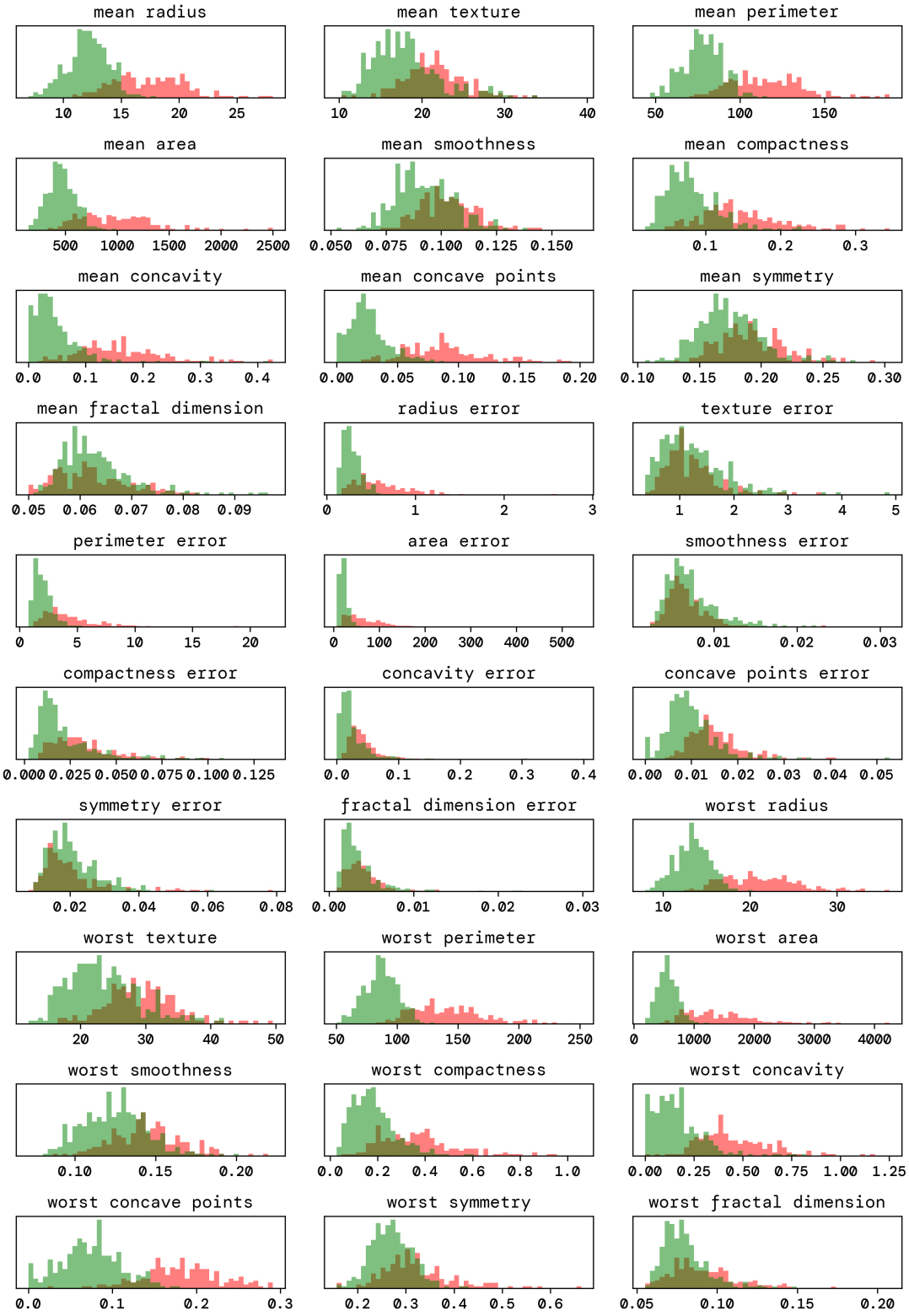

Figure 2: Feature histogram. The red class (0) represents malignant cases, while the green class (1) indicates benign cases.

5. Results and Analysis

In our experiments, we employed the cross-entropy loss function, The gradient of the loss function with respect to the model's parameters was computed using automatic differentiation. We used the SKLearn[8] library to perform cross-validation for a more robust evaluation of our models. We used k-fold cross-validation with $k = 6$ folds. The results presented in this section are based on the average of the results from the 6 folds. The bar plots also show the standard deviation of the results from the 6 folds.

5.1 Hyperparameters

During our hyperparameter tuning phase, we observed interdependencies between certain hyperparameters. Notably, the learning rate and the number of epochs showed a significant correlation, as did the batch size and the learning rate. Such dependencies imply that these hyperparameters cannot be optimized independently for the most effective training process. Ideally, a comprehensive grid search across multiple dimensions of hyperparameters would be conducted. However, due to constraints in time and computational resources, our experiments were limited. We were able to conduct grid searches, but these were restricted to two dimensions at most. While this limitation prevented a thorough exploration of the hyperparameter space, the results from these partial grid searches provided valuable insights. They served as indicators, pointing us towards regions in the hyperparameter space where optimal settings might exist. The analysis of these results suggests that while we have identified promising hyperparameter settings, a more exhaustive search might yield further improvements. We use the same base hyperparameters when we tune the other hyperparameters.

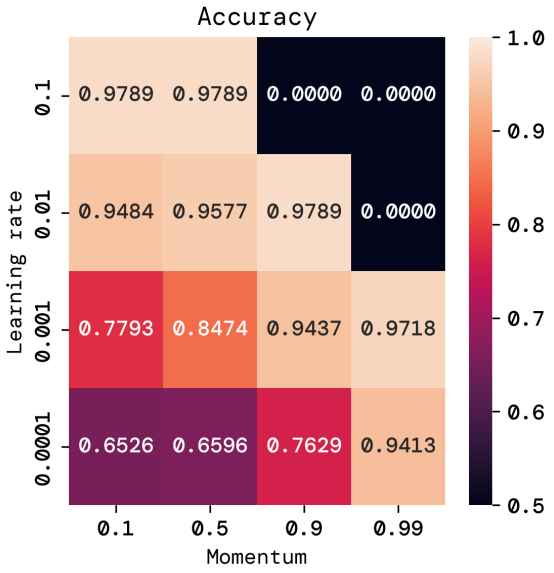

Figure 3: Accuracy versus learning rate and momentum.

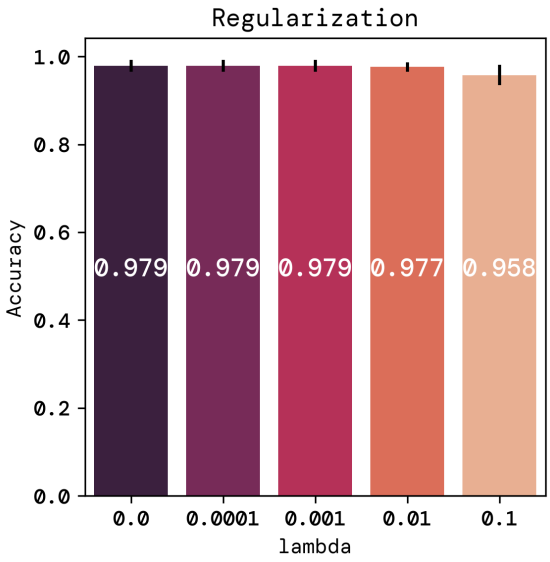

Figure 4: Accuracy versus regualarization.

Our findings indicate that incorporating momentum is beneficial, particularly with a value of around 0.9, which emerged as the most effective in our tests. This momentum allows for a higher learning rate, enhancing the training process. In contrast, excessive regularization seems to adversely affect model performance.

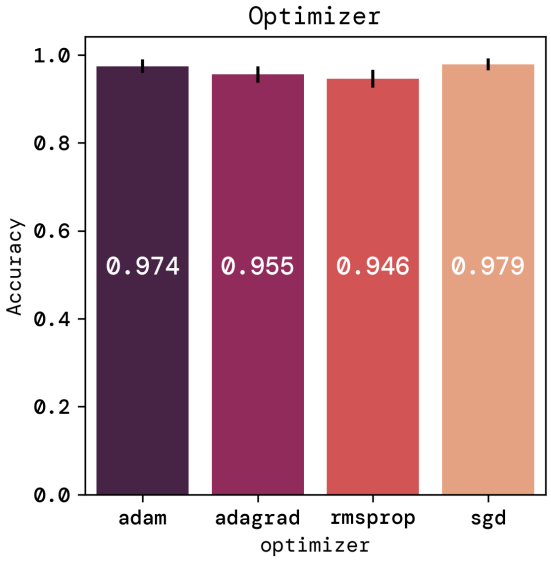

Figure 5: Model accuracy across different optimizers. here sgd is standard gradient descent.

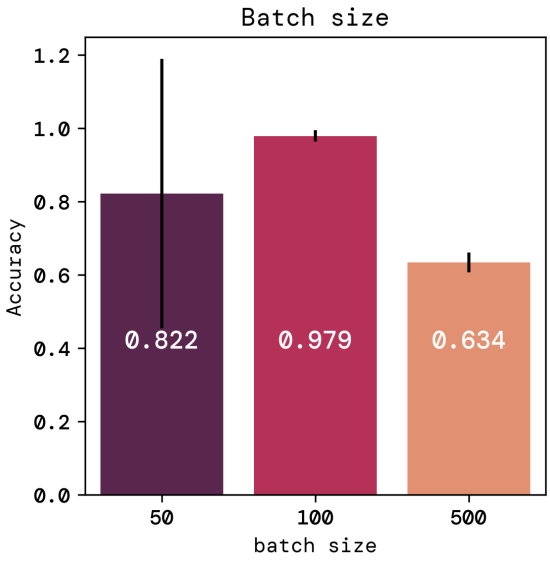

Figure 6: Accuracy based on batch size.

The choice of optimizer did not significantly impact the model's performance. However, Adam and standard gradient decent with momentum showed a slight advantage over others. The batch size also played a role, with certain sizes yielding better accuracy. With a batch size of 50, the model accuracy varied greatly between folds.

In order for our model to correctly classify the data, it is essential that the last layer's activation function is a sigmoid function. This is because the sigmoid function outputs a value between 0 and 1, which can be interpreted as the probability of the data point belonging to class 1. The choice of activation function for the hidden layers did not significantly impact the model's performance. We used one hidden layer with 2 nodes when testing activation functions.

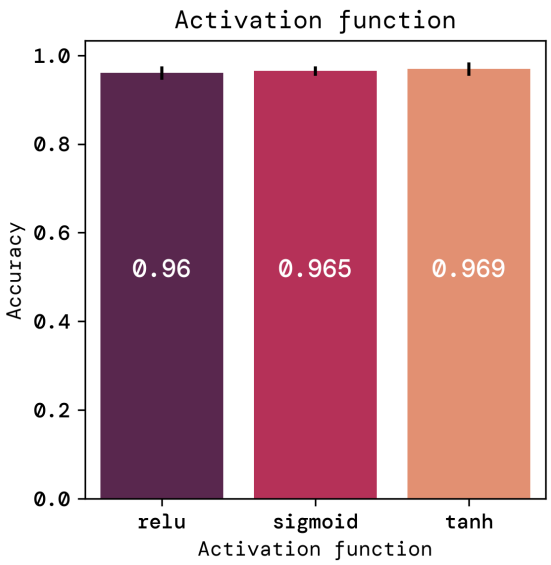

Figure 7: Model performance with different activation functions.

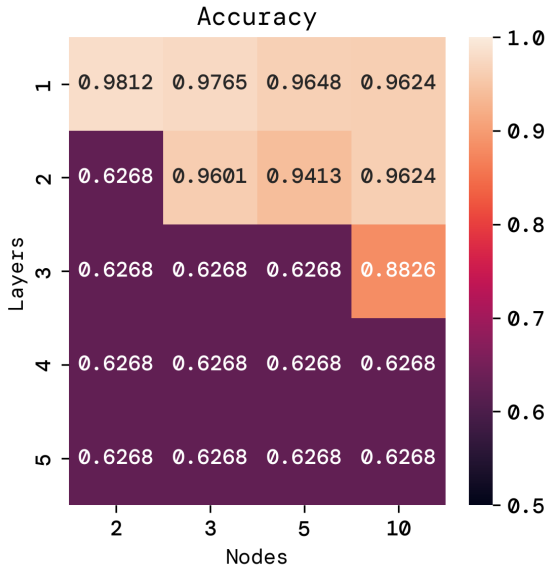

Figure 8: Accuracy in relation to the number of layers and nodes.

5.2 Final Evaluation and Comparisons

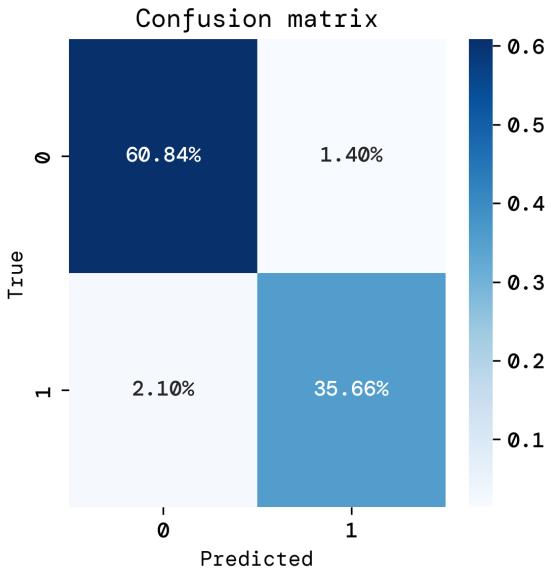

Figure 9: Confusion matrix for our model.

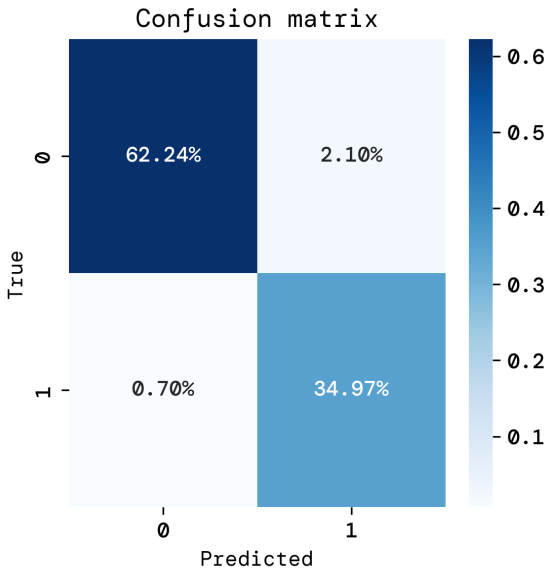

Figure 10: Confusion matrix for SKLearn's model.

In our classification task, logistic regression achieved an accuracy of 96% on the test set, while our neural network model achieved 95%. This performance is comparable to the 96% accuracy obtained with SKLearn's logistic regression model. It's important to consider that this comparison may not be entirely direct due to differences in hyperparameters between the models. Nevertheless, efforts were made to align the hyperparameters as closely as possible. An interesting observation is that increasing the number of layers and nodes in the neural network did not significantly enhance its performance. This suggests the possibility of overfitting to the training data, indicating that a simpler model could be more effective or that additional regularization techniques might be required.

6. Conclusion

This project has delved into the capabilities of neural networks in addressing classification challenges, comparing them with traditional logistic regression models. Using the Wisconsin breast cancer dataset, we found that neural networks, with a classification accuracy of 95%, perform nearly as well as logistic regression models, which achieved 96%. This result is in line with the 97% accuracy achieved using SKLearn's implementation. These findings highlight the competitive nature of neural networks in classification tasks, even when compared to more traditional models. However, they also underscore the importance of careful model selection and hyperparameter tuning in achieving optimal performance. While neural networks offer flexibility and power, they may not always outperform simpler models, especially in cases where the underlying data patterns are not exceedingly complex. Future work could explore further optimization of the neural network architecture and hyperparameters, as well as the application of these models to more diverse datasets to fully assess their generalizability and efficacy in various classification scenarios.

Appendix A

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press.

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

[3] Morten Hjorth-Jensen. Fys-stk4155/3155 applied data analysis and machine learning. `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/intro.html`, 2021.

[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[5] Izaak Neutelings. Tikz code for neural networks. `https://tikz.net/neural_networks/`, 2021. modified to fit the needs of this project.

[6] Michael A. Nielsen. Neural networks and deep learning, 2018.

[7] OpenAI. Chat-gpt. `https://openai.com/chatgpt`, 2023. Some of the code and formulations are developed with the help from Chat-GPT.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Brage Wiseth, Felix Cameren Heyerdahl, and Eirik Bjørnson Jahr. MachineLearning-Projects. `https://github.com/bragewiseth/MachineLearningProjects`, November 2023.

[10] W. Wolberg, O. Mangasarian, N. Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.