Machine Learning Project 4

Anthony Vasquez
Ph:313-920-8877
Email: avasque1@jh.edu

Editor: Anthony Vasquez

Abstract

This document presents performance results of using four networks, a linear perceptron, logistic regression, feedforward, and autoencoder networks to train on several datasets. The trained model performance metrics used are accuracy for logistic regression, mean-squared error (MSE) for the linear perceptron, cross-entropy for feedforward classification problems, MSE for feedforward regression problems, and MSE for the autoencoder.

Keywords: Accuracy, Activation, Autoencoder, Bottleneck, Classification, Cross-entropy (CE), Decoder, Encoder, Feedforward, Hidden Layer, Hyperbolic-Tangent, Input Layer, Linear, Logistic, Loss, Mean-Squared-Error, Nonlinear, Neural Network, One-hot-encoding (OHE), Output Layer, Perceptron, Pretrained, Rectified linear Unit (ReLU), Regression, Sigmoid, Signum, Softmax, Weights

1 Introduction

Neural networks remain the standard in solving optimization and deep learning problems. To explore the underlying concepts that make neural networks, this project explores some of the most well-known networks. In this work, a linear perceptron network is trained on regression data, and a logistic regression network is trained on classification data. In addition, a feedforward network is trained on both regression and classification data, and finally, an autoencoder network is trained and evaluated. **Hypothesis**: Given that the datasets used in these experiments are generally not difficult, it is likely that all neural networks used in these experiments will perform better than 90% for accuracy metrics and less than 1.0 for MSE and CE loss metrics.

1.1 Data Used

The datasets used for classification in these experiments include breast-cancer-wisconsin, car, and house-votes-84. For regression, abalone, forestfires, and machine datasets were used. See the appendix for more information on each of the datasets.

1.2 Data Preprocessing

All datasets underwent standardization by using the equation $z(x) = \frac{x-\mu}{\sigma}$, where x is the data value, μ is the mean, and σ is the attribute standard deviation. Categorical attributes were encoded by using non-binary encoded values prior to standardization. OHE was also used to calculate CE loss for the classification datasets of the feedforward network. OHE will be further explained in the feedforward section.

1.3 Stratified K-fold Cross Validation

The data was separated into several stratified train and validation sets for k-fold cross validation, where k = 5. The full data was stored as a pandas dataframe but was subsequently split into smaller dictionaries for the randomized sampling of the data and each dictionary was appended to a list that was iterated through to obtain the five train and validation sets.

1.4 Model Tuning

A trial-and-error method was used to manually toggle the learning rate and number of training iterations for each model. It was found that a learning rate of 0.001 at 100 training iterations for linear the linear perceptron and logistic regression networks was acceptable. For the feedforward classification, a learning rate of 0.001 with train iterations set to 100 also worked well. For the feedforward regression sets, it was found that a learning rate of 0.01 paired with 150 train iterations. Finally, for the autoencoder, the a learning rate of 0.001 with 150 training iterations was found to be acceptable.

2 Algorithms

This section discusses the machine learning algorithms used throughout this experiment.

2.1 Back Propagation

Back propagation was used in the feedforward network and is essentially calculating what was learned by each layer by calculating the partial derivatives of each layer starting from the last layer in the direction of the output. The backpropagation algorithm can be found in page 288 of the class textbook (Alpaydin, 2020).

2.2 Linear Perceptron

The linear perceptron was the least complex model. In this case, the perceptron to uses a linear function to update the weights. The linear function used for the linear perceptron is (**Equation 1**).

Equation 1: The linear function used in the linear perceptron.

$$y_{linear} = \sum_i^N x_i w_i + b_i$$ (Alpaydin, 2020),

In the linear function, x is a single sample of features, N is the number of samples, and *w* are the weights from each input. The output layer uses the Signum function (**Equation 2**) to assign values to the output seen in

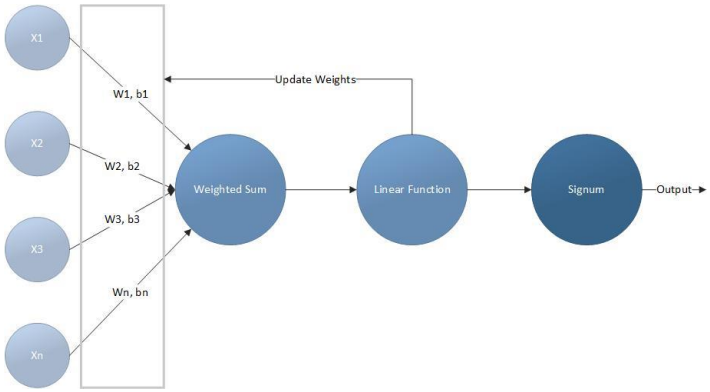
Equation 2 (Alpaydin, 2020). Since the linear perceptron was only used on the regression datasets, the metric used to test the model performance was MSE **(Equation 4**).

Equation 2: The signum function was used for the predictor of the linear perceptron (Signum Function, 2021).

$$Signum = \begin{cases} 1 \; for \; x > 0 \\ -1 \; for \; x < 0 \\ 0 \; for \; x = 0 \end{cases}$$

Figure 1: The linear regression network input layer is represented at the far left. The weights and biases are calculated. A weighted sum is taken and passed to the linear function.

The updates from the linear function are made to the weights and bias before the next iteration. The signum function is used during inference.



2.3 Logistic Regression Network

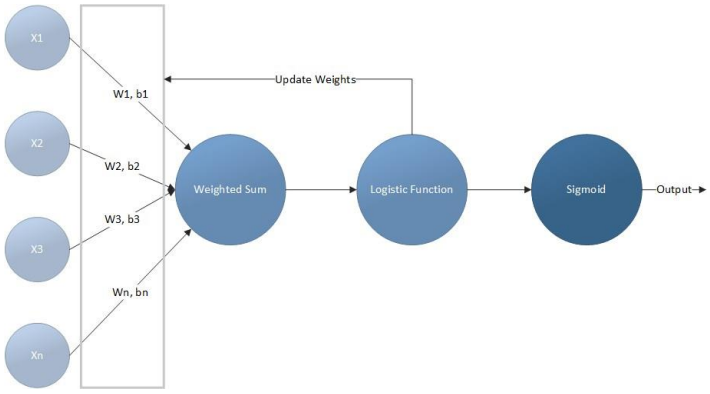
The difference in the architecture of the logistic network and the linear perceptron is that the logistic network uses the logistic function as opposed to the linear function. It should be noted that the linear function is embedded in the logistic function and appears in the exponential function of the denominator of **Equation 3**.

Equation 3: The logistic function is used to calculate the probability of a class given a feature.

$$P(C_1 | x) = \frac{1}{[1+e^{-(w^T x+w_0)}]},$$

In the logistic function, x is a sample vector, w^T are the current weights transposed and w_0 are the initial weights (Alpaydin, 2020).

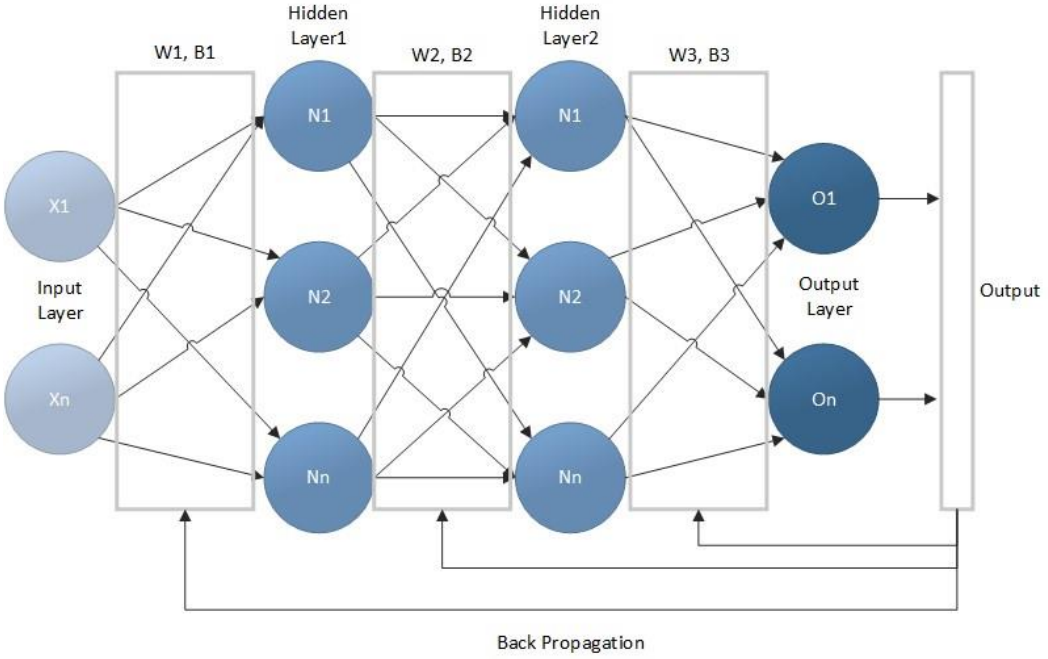
Figure 2: The logistic regression input layer is represented at the far left. The weights and biases are calculated. A weighted sum is taken and passed to the logistic function. The updates from the logistic function are adjustments are made to the weights and biases before the next iteration. This network uses the sigmoid function during inference.



2.4 Feedforward Network

The feedforward neural network is essentially the multilayer perceptron. In the neural network, there are any number of input nodes followed by any number of hidden layers and any number of output layers. The feedforward network implemented here is a network that has an input size of the number of features in the dataset followed by two hidden layers and an output of size one for regression, or of the size of the number of unique classes for classification datasets (**Figure 3**). The feedforward neural network used the Sigmoid (logistic) or hyperbolic tangent activation functions as well as backward propagation of optimization. For inference, it was instructed to use the CE loss combined with the Softmax for the classification problems, and MSE for regression problems. The ReLU activation function was also used to satisfy curiosity, but the only values represented here are using the hyperbolic tangent.

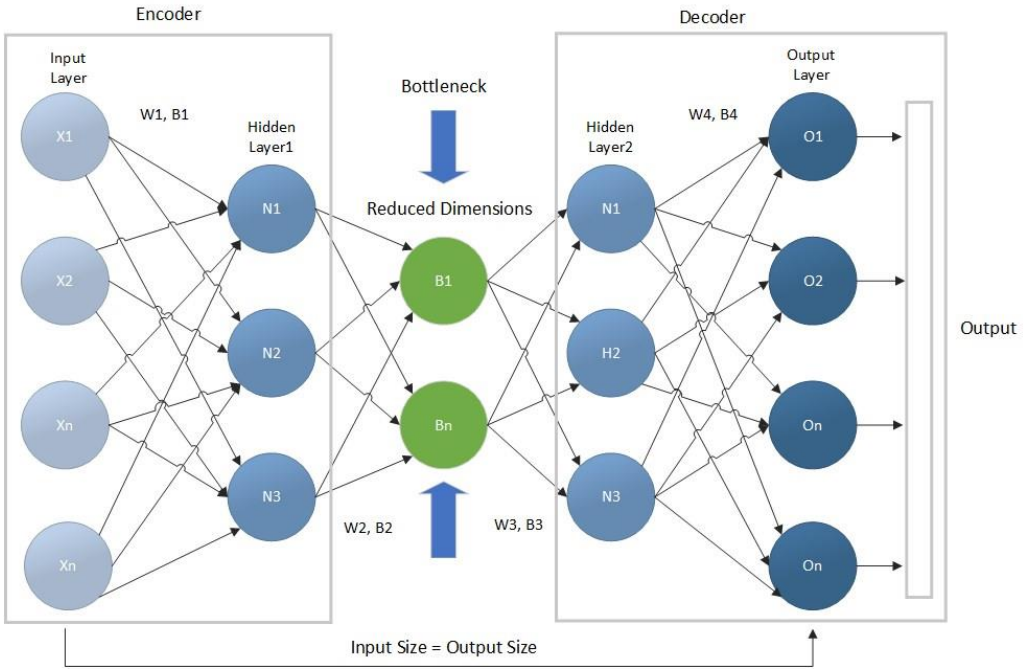
Figure 3: The feedforward network implemented for this project are shown here. The size of the input layer is proportional to the number of dataset features followed by two hidden layers proportional in size to the number of samples. The number of output layers is of size one for regression datasets and of size equal to the number of unique classes in the classification datasets.



2.5 Autoencoder

The autoencoder is a special case of the feedforward network. The way that it was implemented here was to use a feedforward network as the encoder with the input layer the size of the number of features (**Figure 4**). The next layer is about half the size of the input layer, followed by a bottle neck layer of size two. The architecture of the decoder includes an input layer of the size of the encoder output layer, followed by a hidden layer the size of the encoder hidden layer, and finally the output layer the size of the input layer. The autoencoder is an unsupervised algorithm, so class labels where not needed. The output of the autoencoder are learned features of x.

Figure 4: The autoencoder implemented here has an input layer the size of the number of features followed by a smaller layer of about half the size of the input layer which effectively makes the encoder. The next layer is further reduced to two nodes in the bottleneck section of the network. The bottleneck section is really a part of both the encoder and decoder. After the bottleneck layer, the decoder inputs this data and increases the nodes to the size of the encoder's first hidden layer, followed by an output the size of the input of the encoder.



3 Results

This section discusses the metrics used in obtaining the results for each neural network.

3.1 Test Metrics Used

MSE (**Equation 4**) was used to measure the performance of all regression problems and the autoencoder in these experiments.

Equation 4: MSE used for regression and the autoencoder network.

$$mse = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2 \quad \text{(Alpaydin, 2020)}$$

Accuracy (**Equation 5**) was used to test the performance for logistic regression of the three classification datasets.

Equation 5. Accuracy calculation used in this report.

$$accuracy = \frac{correct\ Predictions}{toatal\ samples}$$

Finally, CE (Equation 6) loss was used to test the performance of the feedforward network tested on the classification datasets.

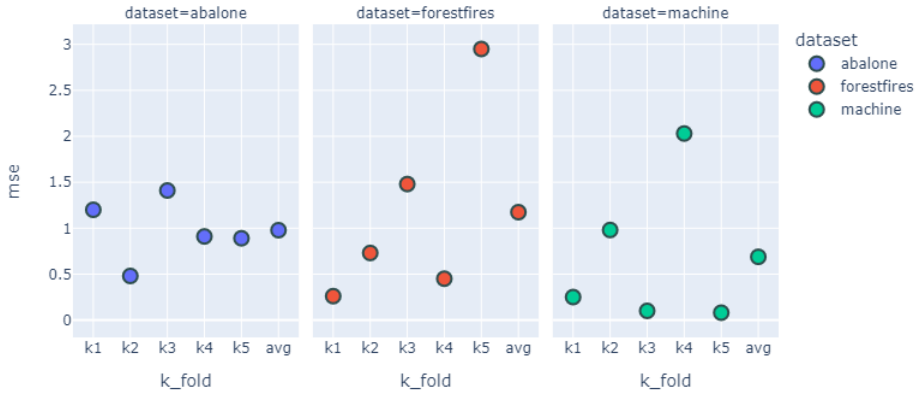
Equation 6: Cross Entropy Loss was used for feedforward classification (Alpaydin, 2020).

$$CE = -\sum_{i=1}^{output\ Size} y_i \cdot log\hat{y}_i$$

3.2 Linear Perceptron MSE Results

The linear perceptron's performance exceeded expectations. On average, it performed below 1.0 for the machine dataset (**Figure 5**). This is the first time that I have seen MSE results this low with this dataset. Abalone also outperformed expectations given the difficulty of the dataset. The linear perceptron generally trained well with respect to the hypothesis value, but resulted in an outlier that moved the average MSE over the 1.0 hypothesis value by a small margin.

Figure 5: The linear perceptron showed better than expected results given the difficulty of the datasets. All the models scored above the hypothesis value for some test sets, but all except the forestfire dataset scored an average MSE below the hypothesis value for MSE score. It is unclear if the lack of network performance for forestfires was the result of under training or overtraining the network. More tests need to be done before committing to an answer.

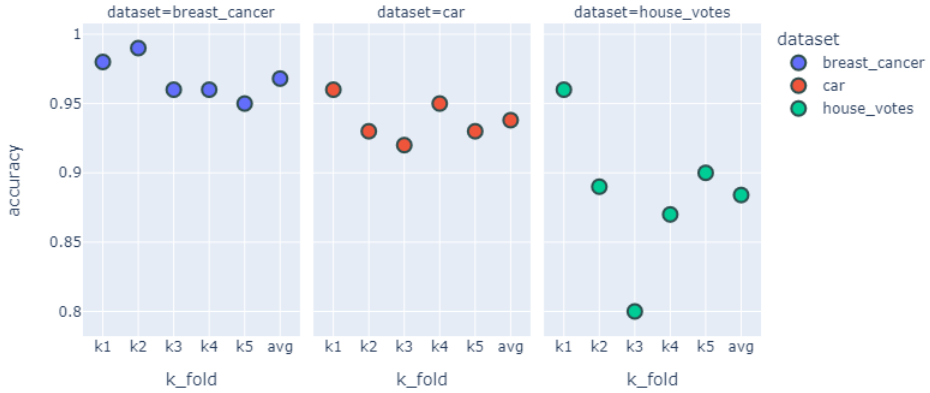


3.3 Logistic Regression Results

The logistic regression models scored above the hypothesized 90% mark for the breast-cancer and car datasets but seemed to struggle with the house-votes dataset (**Figure 6)**. It is possible that the 100 training iterations caused the model to overfit given that it is a small two class dataset. If breast-cancer-Wisconsin was a smaller dataset, a reasonable guess is that it would have also had similar results to house-votes.

Figure 6: Logistic regression scored high accuracy for each dataset but seemed to have fumbled on the house-votes dataset, which is a surprising result given the small size and complexity of the dataset. The reason for the mediocre performance on house-

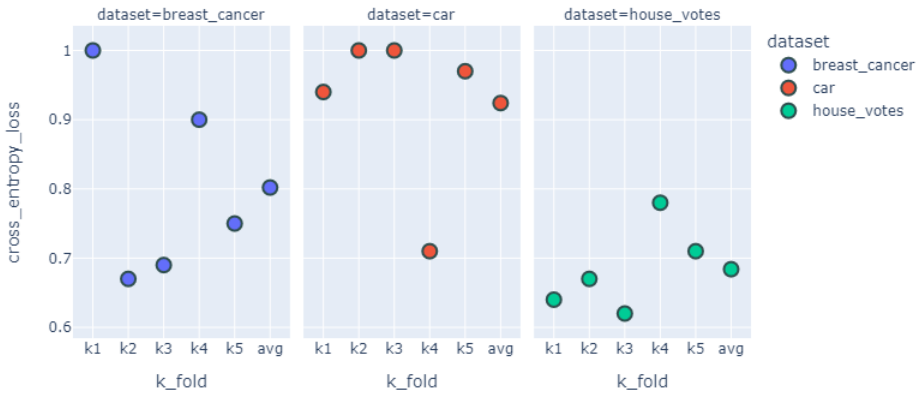
votes could reasonably be due to an overtrained model. More tests need to be done to provide accurate answers.



3.4 Feedforward Network Cross Entropy Results

It is hard to say how well the feedforward models performed on the classification datasets. The best score was on the house-votes dataset, but all scored at one or below the hypothesized value for CE datasets (**Figure 7**).

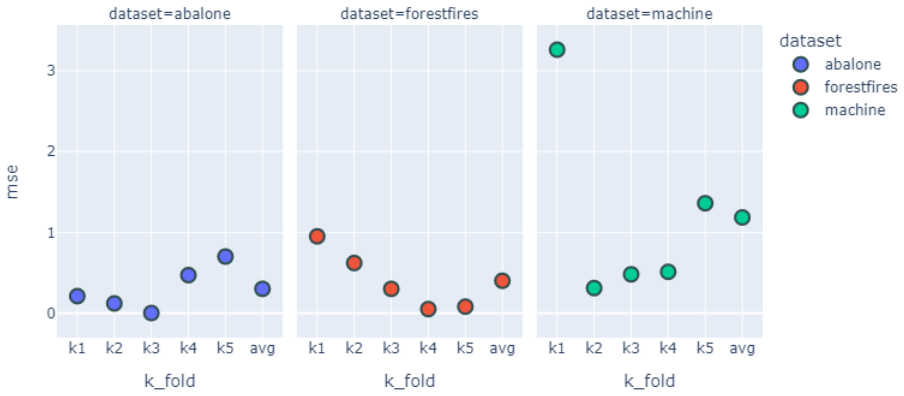
Figure 7: The feedforward network performed at or below the hypothesized value for all the classification datasets.



3.5 Feedforward Network MSE Results

The feedforward network for regression performed better than the linear regression network (**Figure 8**). The increased performance is most notable when comparing the forestfire and abalone datasets. The feedforward network performed about the same as linear perceptron on the machine datasets. Both networks showed a scattered pattern that was largely varied.

Figure 8: The feedforward network for regression datasets performed better or about the same when compared to the previous models tested.



3.6 Autoencoder MSE Results

The autoencoder was a special case of the feedforward network or sections of two feedforward networks is more accurate. No labels were used and the MSE calculation was done by setting the encoder input layer as the truth data and setting the decoder output to the predicted data. The variable swap made it easy to plug the values from each network into the MSE equation (**Figure 9**) once training was finished. The results of the autoencoder are shown below.

Figure 9: The MSE was calculated for each of the datasets for the autoencoder. The car dataset did not perform well in comparison to the other tested datasets. The MSE scores for the autoencoder are not as good as some of the previous architectures. It is important to note that the MSE should not be compared to the other architectures, because truth labels are swapped and replaced by feature data, and the predicted label become the decoder's output.

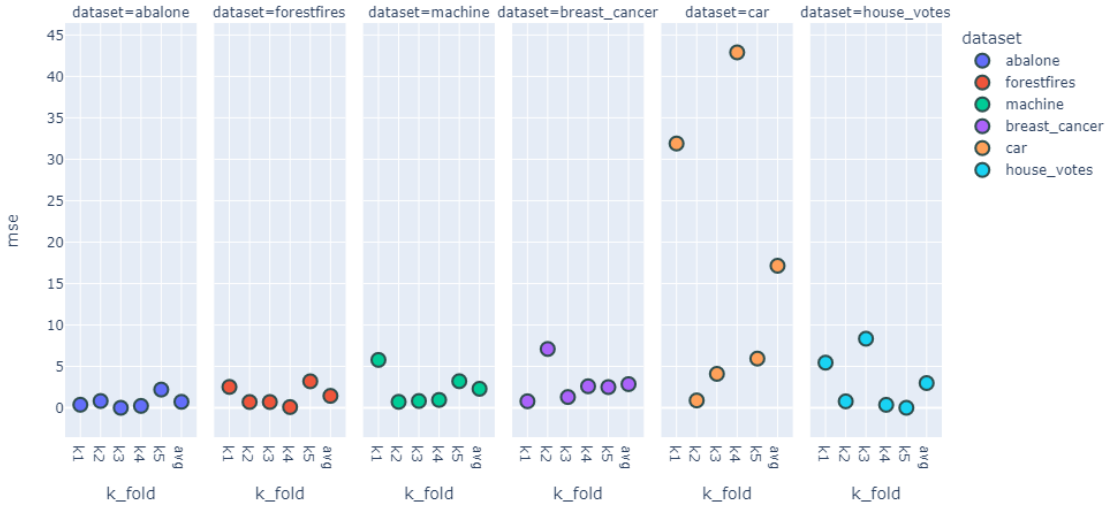


Figure 10: The heatmap of the encoder input layer is shown below. The x-axis are the feature vectors, and the y-axis are node samples. The color embedded cells represent the feature values at each cell.

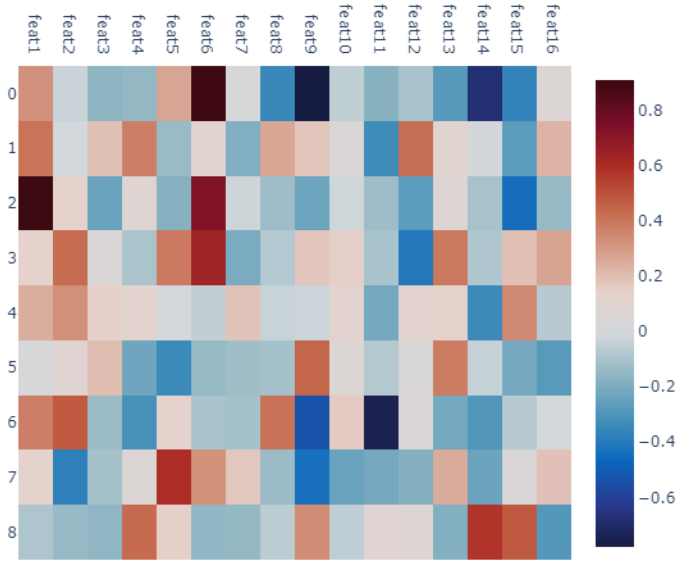


Figure 11: The decoder's output features are shown as a heatmap. The x-axis represents the output of the decoder and is color embedded to show the feature value.

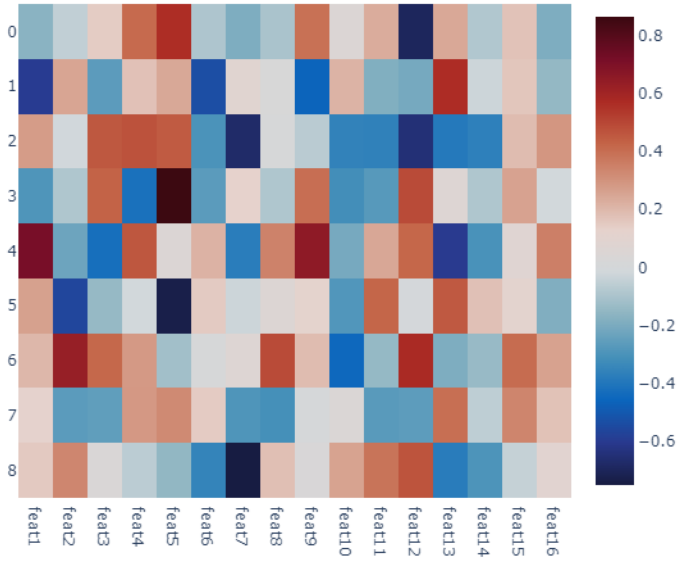
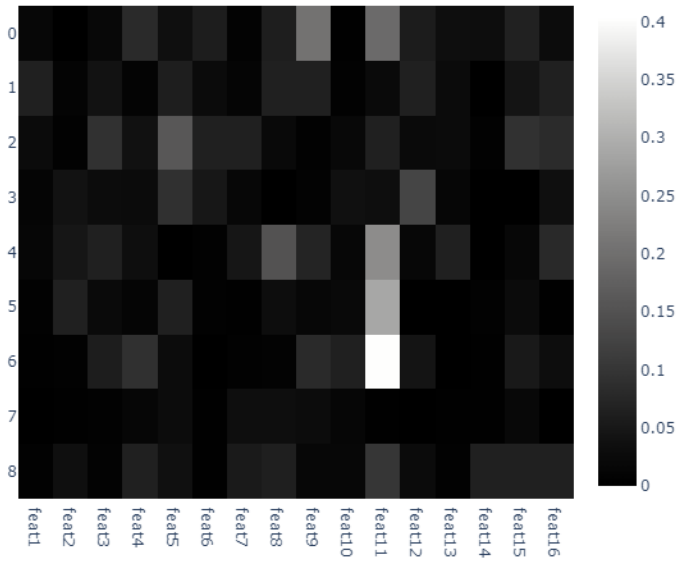


Figure 12: The MSE was calculated for each individual cell and was plotted as a heatmap to show that the input values are very close to the decoder output. Here the x-axis are the features, and the y-axis are reduced sample features. The color embedding

represents the MSE of each cell. The darker the cell, the less the difference is between the encoder input and the decoder output.



4 Conclusion

The study of neural networks has been the hardest subject in this class. It was both time consuming and mind bending. I would personally take this assignment over the different decision trees any day. Decision trees were not as complicated, but the amount of bookkeeping made it hard to maneuver and reuse code. The last sentence is a dig on me more than on decision trees. Neural networks are my favorite subject thus far in the class, which made it well worth the hard work and effort.

References

Alpaydin, E. (2020). In E. Alpaydin, *Introduction to Machine Learning.* Cambridge: The MIT Press.

Marko Bohenec, B. Z. (1997, June). Car Evaluation Database. Avignon, France.

Paulo Cortez, A. M. (2007, December). Forest Fires. Guimaraes, Portugal.

Phillip Ein-Dor, J. F. (1987, October). Relative CPU Performance Data. Tel Aviv, Israel.

Schlimmer, J. (1984). 1984 United States Congressional Voting Records Database. Washington D.C., USA.

Signum Function. (2021, 11 12). Retrieved from CUEMATH: https://www.cuemath.com/algebra/signum-function/

Waugh, S. (1995). Extending and benchmarking Cascade-Correlation. Hobart, Tasmania, Australia.

Wolberg, W. H. (1992, July 15). Wisconsin Breast Cancer Database. Madison, Wisconsin, USA.