NN Classification for Targeted Advertising
EN.605.647.8VL.FA25

Table of Contents

Author: Andrew Saifnoorian

Executive Summary

This project implements and evaluates two neural network architectures for classification. The networks are designed to address a crucial aspect in advertising campaigns which is to determine whether the campaign will be worthwhile. We want to create a network that can correctly assess whether the targeted campaign will provide a positive or negative return. Our model considers two factors: the 'Size of Wallet' (SOW) and the 'local affluence code' (LAC). SOW will capture a household's disposable income, with a higher value representing greater available income. LAC, on the other hand, is a composite metric that combines multiple datapoints like housing quality and neighborhood statistics. A higher LAC score portrays better housing conditions and neighborhood attributes. The output will be a targeted advertising code assignment (TACA) which tells us whether the campaign was successful and had a positive return, or it was unsuccessful and had a negative return. If we were to simply guess the pattern, we would think that larger SOW and higher LAC would mean that there would be more money to spend and we should target advertisements towards those households. However, by using these inputs with our network, we will have the ability to quickly discover underlying trends and patterns with the inputs which associate to TACA scores in a scalable and robust manner. The goal of this network is to provide high classification accuracy in identifying worthy households to invest a targeted advertisement in, which will also function as a cost-effective and rapid tool that will be able to replace expensive, data-intensive manual analysis. This will maximize profits as revenue per amount spent should increase with the help of our network.

Key Findings

- Both neural network architectures achieve identical performance indicating that there may be an approximate linear relationship between inputs (SOW & LAC) and output (TACA)
- Both models exhibit conservatism with a higher chance of being correct when marking negative returns (0 TACA score)
- Limited training dataset hinders our ability to make a final call on which model works best with the multi-layer neural network having 9 parameters but only ten datapoints to train on

Problem Description, Network Design, and Solution Approach

Problem Description

Our objective is to predict whether a targeted advertising campaign directed at specific households will yield a positive financial return. The prediction is based on two socioeconomic indicators associated with each household. Two inputs, SOW & LAC, are being used to predict the TACA score, which will be 1 when the expected return on advertising investment is positive and 0 otherwise. The classification task is inherently challenging due to the nonlinear relationships that may exist between household affluence indicators and advertising responsiveness.

The practical significance of this problem lies in optimizing advertising expenditure. Accurate classification enables organizations to allocate marketing resources toward households with the highest probability of favorable response, thereby maximizing return on investment while minimizing wasted expenditure on non-responsive segments.

Data Characteristics and System Inputs

The dataset comprises 20 labeled samples, each representing a distinct household. Two continuous input features characterize each sample. The Local Affluence Code (LAC) quantifies neighborhood-level economic indicators, including housing values, structural quality, and crime rates, scaled to the interval [0, 3]. The Size of Wallet (SOW) estimates household disposable income based on gross income, household composition, and demographic factors, also scaled to [0, 3].

Data partitioning followed a deterministic scheme: odd-numbered samples (rows 1, 3, 5, ..., 19) constituted the training set comprising 10 samples, while even-numbered samples (rows 2, 4, 6, ..., 20) formed the testing set of equal size. The training set exhibited balanced class distribution with 5 positive and 5 negative instances. The testing set contained 3 positive and 7 negative instances, introducing a class imbalance that affects performance metric interpretation.

Network Design

Model A implements a *single-layer perceptron* architecture representing the simplest form of neural classifier. The network topology consists of two inputs (LAC & SOW) connected directly to a single output neuron. Each input connection carries an adaptable weight coefficient, and the output neuron incorporates a bias term. The weighted sum of inputs plus bias passes through a sigmoid activation function, producing a continuous output in the interval (0, 1) that represents the estimated probability of class membership.

The single perceptron computes the output as σ ($w_1 \cdot$LAC + $w_2 \cdot$SOW + b), where σ denotes the sigmoid function, w_1 and w_2 are input weights, and b is the bias. This architecture defines a linear decision boundary in the two-dimensional input space. The model's representational capacity is therefore limited to problems that are linearly separable or approximately so.

Model B employs a *multi-layer perceptron* (MLP) with a single hidden layer to introduce nonlinear classification capability. The architecture follows a 2-2-1 topology: two input (LAC and SOW) values, two neurons in the hidden layer, and one output neuron. All neurons in the hidden and output layers utilize sigmoid activation functions. The network comprises four weight parameters in the input-to-hidden connections, two hidden layer biases, two hidden-to-output weights, and 1 output bias, totaling nine adaptable parameters.

The hidden layer enables the network to learn nonlinear decision boundaries through composition of sigmoid functions. Each hidden neuron effectively creates a soft linear separator, and their combined outputs enable the network to approximate arbitrary decision regions. The two hidden neurons provide sufficient capacity to capture potential nonlinear relationships between affluence indicators and advertising response patterns present in the training data.

Training Strategy & Optimization Approach

Both networks were trained using online stochastic gradient descent, wherein weight updates occur after presentation of each individual training sample rather than after batch accumulation. This approach was mandated by the assignment specifications and follows the feedforward-backpropagation (FFBP) cycle methodology. During each training epoch, all ten training samples were presented sequentially, with weights and biases adjusted after each sample based on the instantaneous error gradient. Training proceeded for exactly 30 epochs as specified.

The mean squared error (MSE) between network outputs and target TACA values served as the optimization criterion. Weight initialization followed a uniform distribution, ranging from -1 to 1. Bias terms were similarly initialized using uniform random sampling. To mitigate sensitivity to initial conditions, 20 independent training runs were executed for each architecture, each with different random weight initializations. The model achieving the lowest final training MSE was retained for subsequent evaluation.

Classification Decision Logic and Threshold Optimization

The sigmoid output of each trained network produces continuous values in (0, 1), requiring a threshold logic to generate discrete binary classifications. A classification threshold τ partitions the output space: samples with output $\geq \tau$ are classified as positive (TACA = 1), while outputs $< \tau$ yield negative classifications (TACA = 0). The threshold value critically affects the trade-off between sensitivity and specificity.

Threshold optimization was conducted by evaluating classification performance across thresholds ranging from 0 to 1, with 0.05 increments. For each threshold, the confusion matrix was computed, yielding true positive (TP), true negative (TN), false positive (FP), and false negative (FN) counts. Derived metrics including sensitivity (TP rate), specificity (TN rate), positive predictive value, negative predictive value, and overall accuracy were calculated. The Receiver Operating Characteristic (ROC) curve was constructed by plotting sensitivity against (1 - specificity) across all thresholds, with the Area Under the Curve (AUC) computed via trapezoidal integration.

Decision-Support Functionality

The dual-model approach provides a comparative framework for evaluating classifier complexity against predictive performance. The single perceptron establishes a baseline representing the performance achievable with linear decision boundaries. If the relationship between affluence indicators and advertising response is approximately linear, the perceptron should achieve competitive performance with minimal computational overhead and reduced risk of overfitting.

The multi-layer perceptron extends this baseline by introducing nonlinear representational capacity. If the true decision boundary exhibits curvature or disjoint regions in the LAC-SOW space, the MLP's hidden layer provides the necessary flexibility to capture these patterns. The comparative analysis of both architectures on identical training and testing partitions enables direct assessment of whether the additional complexity of the MLP translates to improved generalization performance on unseen data.

From a decision-support perspective, these networks function as rapid screening tools. Once trained, classification of new households requires only a forward pass computation, enabling real-time or batch processing of large household databases. The probabilistic interpretation of sigmoid outputs further allows confidence-weighted ranking of households, enabling graduated marketing strategies beyond binary targeting decisions.

Model Evaluation & Analysis

Multiple metrics were used to help determine the optimal model, including accuracy, mean square error (MSE), specificity, and sensitivity. Each metric was compared, and an overall scoring metric was determined for the best model. Both the single and multi-layer approaches adopted an online gradient descent method with sigmoid activation. As for the dataset, we used the odd rows for training and the even ones for testing purposes.

For each experiment, the network completed 30 cycles through the training set, with the learning rates being parametrized. The cycles were completed 10 times for random initial weights and biases. For each run, the total combined Big E was computed for each set of initial weights, and the weights with the smallest Big E were chosen to train and analyze the model. Afterward, the same learning rates were parametrized, and the score thresholds for the output node were parametrized, and the accuracy metrics were computed for each possible combination of parameters. The results were then analyzed to determine the optimal model which would be used for the testing dataset.

Finally, we developed the following scoring equation that incorporates the most important evaluation criteria from our understanding. The model would have good overall classification accuracy, but also scoring accuracy, and a good balance between specificity and sensitivity.

$$Model\ Score = Accuracy + \ 0.5 * (sensitivity + specificity) - Mean\ Squared\ Error$$

Model A: Single Perceptron Network

Figure A1 in the appendix shows how we tested the different levels of learning rate (eta) and threshold to find the highest accuracy. As shown, the region around a 0.25 to 0.5 threshold and higher learning rates, the overall model accuracy was highest. We also analyzed the mean squared error across various values of eta and saw a steady decline, as shown in figure A2, as the learning rate (eta) increased from 0 to 1.

As noted earlier, we used the 'Model Score' criteria to get a combined score of all four metrics, ensuring the model would have good overall classification accuracy, but also accurate scoring, and likely a good balance between specificity and sensitivity. After computing the score for each model, the highest scoring single neuron model was found to have a learning rate (eta) of 1.0, and a classification threshold of 0.25. The model produced an overall score of 1.6427 with an MSE of 0.15.

The final prediction equation, under the model was:

$$Predicted\ TACA = (-1.4596) * LAC + (-2.2664) * SOW + 3.6285$$

We then took the model above and tested it against our test dataset. The model performance summary is shown in Table A1.

Analyzing the Single Neuron model results, at a high level the model scores relatively high in accuracy (0.8) and is strong at determining households that are a negative for target advertising campaigns (based on high specificity rate of 1). One downside of the model, however, is that it has a smaller sensitivity rate (0.33), as it labels too many households as negative, even ones that are true positive. Consequently, the model is highly likely to be correct about scoring a household as a positive, but this comes at the cost of over-labelling households as negative. An advertising company that has more limited resources would find this model useful in order to be confident that the households it uses targeted advertising on are generating a positive return. If the advertising company had much larger resources, however, and didn't have to be as selective about finding true positives, then a different model that has a higher sensitivity and can identify more of the true positives would be needed. Lowering the threshold of the current model, working with different learning rates, or utilizing training methods would help find a model with higher sensitivity.

Model B: Multi-Layer Network

We applied the same threshold logic, eta values and iterations for our multi-layer neural network as well and used the same criteria for evaluation and comparison. As we built out the model, we compared mean squared errors for different eta values. As shown in figure B1, MSE values stayed relatively high for most eta values, declining significantly once eta crossed 0.7, reaching a minimum value at a learning rate of 0.9 before jumping back up when we eta equaled 1. The lowest MSE, achieved at 0.9, was about 0.18.

We also compared accuracy across different eta and threshold values to see as to which ones provide the highest accuracy. This is shown in figure B2 where we see best performance at eta ranges 0.75 to 0.95 with threshold range hovering around 0.45. It is worth noting that high levels of accuracy were only achieved in this very small region, as evident from the graph, whereas accuracy remained low across most of the other regions.

The model score, for the best model, came out to be 1.27. The final model weights associated with the hidden and outer layer nodes along with the biases are shown in table B1. From this table, hidden layer node 1 shows strong positive weights for both LAC & SOW while having a negative bias. This node most likely has a high activation (i.e., fires) when we have high affluence households with strong income. Hidden node 2 on the other hand shows an opposite impact with a negative weight associated with LAC and positive SOW (with a negative bias as well). This node most likely activates for houses with high disposable income, but lower neighborhood affluence.

Finally, even with this more complicated approach, the best performing model was still at an accuracy of 80% with the sensitivity and specificity measures also identical to that of the single layer neuron. It is worth mentioning that the multi-layer network comes with an increased computational cost complexity, which should be justified through gains in performance metrics.

Model Comparison

As discussed, the multi-layer network added 6 more parameters when compared to the single layer model. The model was introduced with the aim of capturing a non-linear representation and hence increasing model performance. However, both models achieved 80% accuracy, with a 33.33% sensitivity and 100% specificity. This suggests that it is likely that LAC & SOW have a linear relationship with TACA that is being captured well by the simpler linear model. However, the limited training data hinders our ability to make a final judgment. The current set of data is insufficient for the multi-layer network to meaningfully learn the non-linear relationship. And this is showing in evaluation results whereby mean squared errors are generally higher across all eta values for multi-layer model. MSE is also the driving factor behind the worse 'Model score' of 1.27 for the multi-layer model and may also be the reason behind a more limited optimal hyperparameter region (Figure B1).

Potential Improvements

There are multiple actions that could be taken to improve the single neuron model in the future. One action that could be taken to improve the single-neuron model would be to adopt a more robust and repeatable data-splitting strategy for evaluation. This would potentially improve the model because it reduces the chance that reported performance is based on a single lucky or unlucky split and gives a clearer picture of how the model behaves across different subsets of the data. This could be done by using stratified train/test splits so class proportions are preserved and performance is measured many times. The results could be summarized with an average and a measure of variability so that the accuracy and stability of the model could both be measured. A final testing split could still be preserved if needed, to determine one final performance score on unseen data.

Another action that could be taken to improve the single-neuron model would be to formalize the hyperparameter search and make model selection more reproducible. Candidate settings could be evaluated across several runs so that choices reflect consistent behavior rather than a single best score from one run. The final configuration could then be chosen based on its repeatable performance across these runs, and the number of variations per parameter could also be increased to potentially find a more precise set of optimal parameters.

Lastly, an action that could be taken to improve the single-neuron model would be to employ data preprocessing methods. Before training and evaluating the model, we could standardize inputs, introduce a small set of simple, interpretable feature transforms, and address class imbalance during training. Standardization would make training less sensitive to scale, modest feature transforms (for example, an interaction or absolute difference) could provide additional signal while keeping the model architecture unchanged, and imbalance handling (via weighting or modest oversampling within training splits) could reduce bias toward the majority class. Finally, calibrating the model's output scores on held-out data could help ensure that chosen thresholds produce predictable sensitivity and specificity in practice.

As for the multi layered network, the narrow hyperparameter regions (figure B2) mean that the model is quite sensitive to configuration choices when compared to the single layer network. Hence, more careful tuning and a more robust validation process will be needed for model maintenance and updates. Additionally, we have nine trainable parameters for the network, but only ten datapoints in the training set. Hence, we will not be able to generalize well and risk overfitting the training data. We can overcome this by getting more data, and using the sampling and stratification techniques discussed earlier for the single layer network.

Conclusion

If we are to have a situation where data is always limited, we are better off using a single layer network that provides a linear separator. When compared to the multi-layer network, it delivers similar performance with a third of the parameters. It also has a lower computational cost and the linear relationship makes it easily interpretable for business users as well. Only if we have a large amount of data can we properly implement the multi-layer network and better understand model performance to accurately see if the more complex network adds meaningful value to our analysis.

Appendix

Figure A1: Single Neuron Model Training Dataset Accuracy (Parametrized)

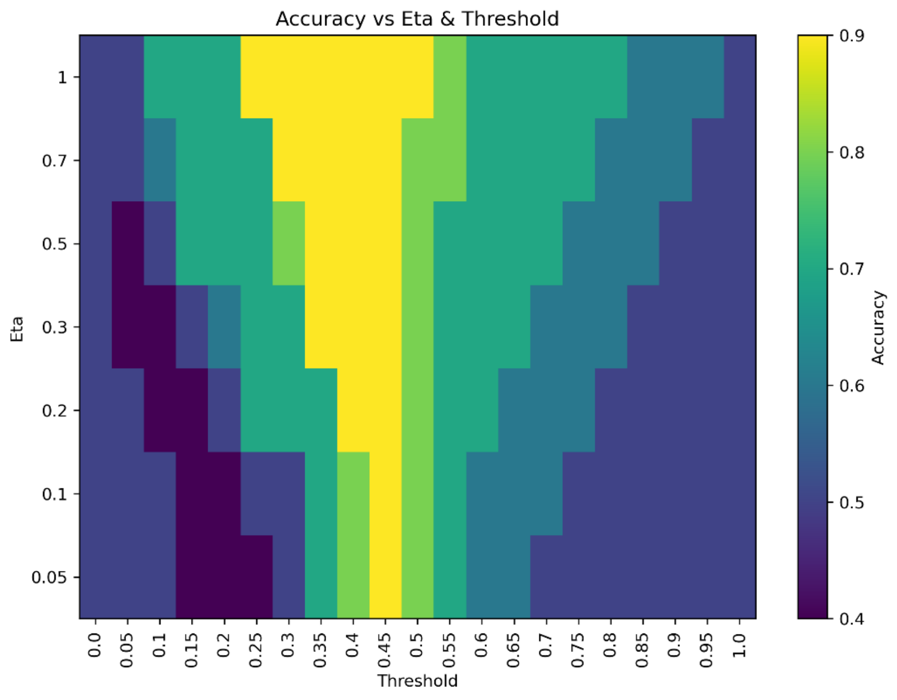

Figure A2: Single Neuron Model Training Dataset Learning Rate & MSE comparison

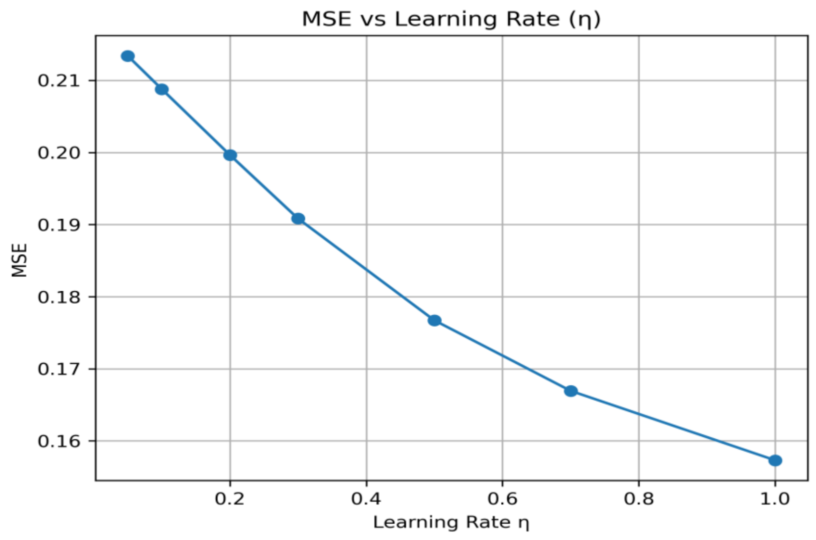

Table A1: Single Neuron Model Testing Dataset Results

Accuracy	**Sensitivity**	**Specificity**	**True Positives**	**True Negatives**	**False Positives**	**False Negatives**
0.8000	0.3333	1.0000	1	7	0	2

Figure B1: Multi-layer Neuron Model Training Dataset Learning Rate & MSE comparison

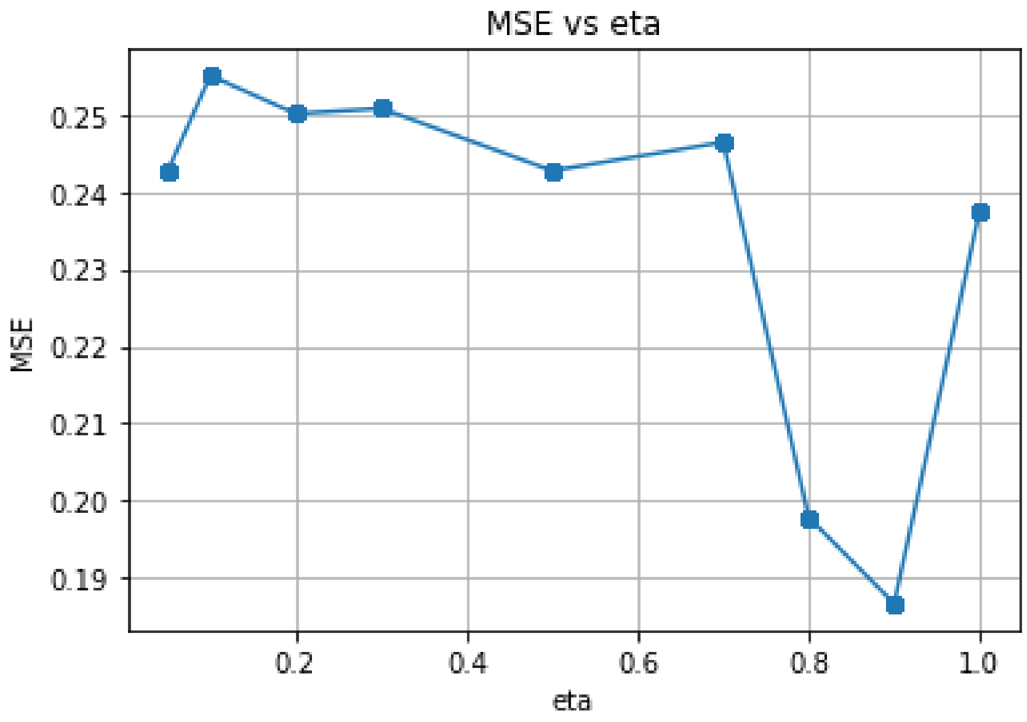

Figure B2: Multi-Layer Neuron Model Training Dataset Accuracy (Parametrized)

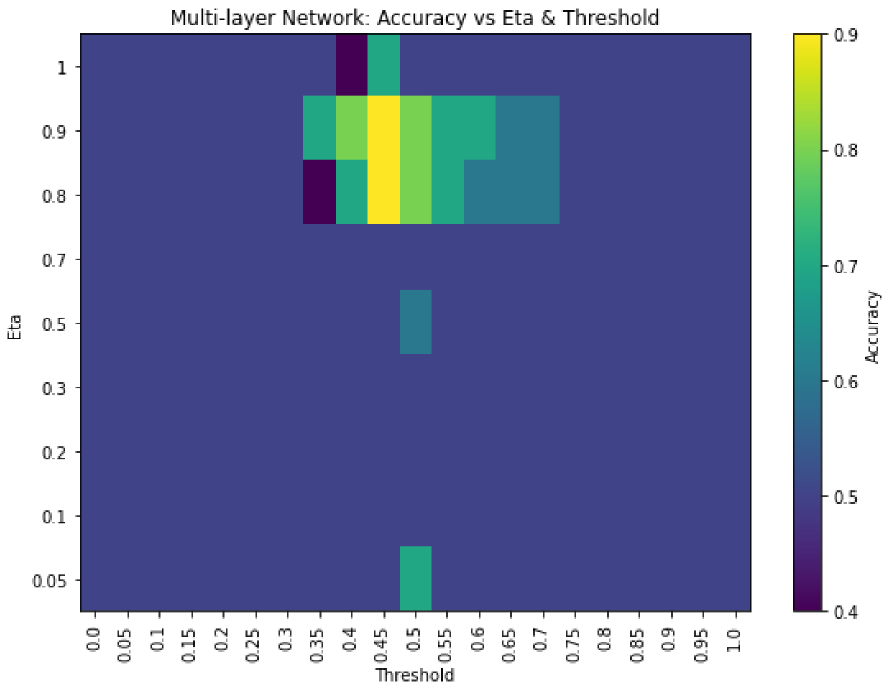

Table B1: Multi-Layer Neural Network Final Weights & Biases

Node	**Weight 1**	**Weight 2**	**Bias**
Hidden Layer 1	0.936744	1.533889	-1.677211
Hidden Layer 2	-0.784491	0.543519	-0.629525
Outer Layer	-2.300592	-0.099356	1.469925

Table B2: Multi-Layer Neuron Model Testing Dataset Results

Accuracy	**Sensitivity**	**Specificity**	**True Positives**	**True Negatives**	**False Positives**	**False Negatives**
0.8000	0.3333	1.0000	1	7	0	2