**Report on Neural Network Models**

**Introduction**

In this report, we explore the optimization and evaluation of neural network models for a classification task as part of the Deep Learning Challenge. Our primary goal is to improve the performance of the model through various hyperparameter adjustments and network architectures. The models are evaluated on their ability to accurately classify data based on predefined features. This report will cover the different configurations used, the performance of each model, and potential alternatives for solving the same problem.

**Process and Methodology**

**Initial Model Setup**

**Starter\_Code\_final.ipynb**

* **Model Architecture:**
  + Hidden Layers: 80 and 30 nodes
  + Output Layer: Sigmoid activation

The initial model setup involves defining the number of hidden nodes and layers in the neural network. We used 80 and 30 nodes in the hidden layers with a sigmoid activation function for the output layer to predict binary outcomes.

A screenshot of a computer program

Description automatically generated

**Model Optimization**

**AlphabetSoupCharity\_Optimization\_1.ipynb**

* **Hidden Nodes: 60 and 40**
* **Output Layer: Sigmoid activation**

For this model, we optimized the network by changing the number of hidden nodes to 60 and 40. This adjustment aimed to see if fewer nodes would enhance the model's generalization capabilities.

A screenshot of a computer program

Description automatically generated

**AlphabetSoupCharity\_Optimization\_2.ipynb**

* **Hidden Nodes: 90, 60, and 30**
* **Output Layer: Sigmoid activation**

This model included an additional hidden layer, making it a three-layer network with 90, 60, and 30 nodes. The purpose was to evaluate whether a deeper network structure would lead to better performance.

*A screenshot of a computer program

Description automatically generated*

**AlphabetSoupCharity\_Optimization\_3.ipynb**

* **Hidden Nodes: 70 and 50**
* **Output Layer: Sigmoid activation**

The final optimization involved adjusting the hidden nodes to 70 and 50. This configuration aimed to balance model complexity and performance, testing a different combination of nodes.

*A screenshot of a computer program

Description automatically generated*

**Analysis**

**Model Performance**

**Evaluation Results**

1. **Starter\_Code\_final.ipynb** 
   * Loss: 0.5705
   * Accuracy: 72.36%

*A screenshot of a computer code

Description automatically generated*

1. **AlphabetSoupCharity\_Optimization\_1.ipynb**
   * Loss: 0.5742
   * Accuracy: 72.36%

*A screenshot of a computer code

Description automatically generated*

1. **AlphabetSoupCharity\_Optimization\_2.ipynb**
   * Loss: 0.5858
   * Accuracy: 72.62%

*A screenshot of a computer code

Description automatically generated*

1. **AlphabetSoupCharity\_Optimization\_3.ipynb**
   * Loss: 0.5762
   * Accuracy: 72.71%

*A screenshot of a computer code

Description automatically generated*

The results show that the accuracy of the models varied slightly. The final model, AlphabetSoupCharity\_Optimization\_3.ipynb, achieved the highest accuracy of 72.71%, compared to the initial model’s 72.36%. This indicates that adjustments in the number of hidden nodes and layers contributed to marginal improvements in performance.

**Performance Analysis**

* **Impact of Hidden Nodes:** Increasing or decreasing the number of hidden nodes impacted the model’s performance. The model with three hidden layers (90, 60, 30) showed a slight improvement in accuracy, suggesting that more complex architectures can capture more intricate patterns in the data.
* **Model Complexity vs. Performance:** The simplest model (60 and 40 nodes) did not perform as well as more complex models. This suggests that additional layers and nodes may help in learning more from the data but also require careful tuning to avoid overfitting.

**Conclusion**

The neural network models evaluated demonstrate a range of performance outcomes based on the number of hidden nodes and layers. The best-performing model was AlphabetSoupCharity\_Optimization\_3.ipynb, with 72.71% accuracy, showing that a balanced approach to network complexity can yield improved results.

**Alternative Approaches**

While the neural network models showed promising results, exploring other machine learning algorithms such as Random Forest or Support Vector Machines could offer different insights. These models might handle the classification task differently and could potentially improve overall performance by addressing data characteristics that neural networks might not capture as effectively.

**Recommendations**

For future improvements, consider:

* **Further Hyperparameter Tuning:** Explore additional configurations and hyperparameters.
* **Cross-Validation:** Implement cross-validation to ensure robust model performance.
* **Feature Engineering:** Investigate additional features or transformations that might improve model accuracy.

**Appendix**

* **Model Code Screenshots:** Included throughout the document.
* **Evaluation Metrics Screenshots:** Included throughout the document.