Assignment 8; CS 158: Machine Learning
Bleecker Coyne

1)
First, I tested with the data from the figure from the assignment to ensure everything was working properly.
TEST on Figure from Assignment:

```
Node outputs (Figure 2):
  h1 = 0.26894
  h2 = 0.23691
  y  = 0.59869
Final weights (Figure 3):
Hidden weights:
       x1          x2          bias
h1: [-0.69724,  1.60275, -1.797243]
h2: [ 0.031795, 0.60179, -1.3982]
Output weights:
    v1          v2          bias
[-1.096313, -0.596670, 1.794853]
```

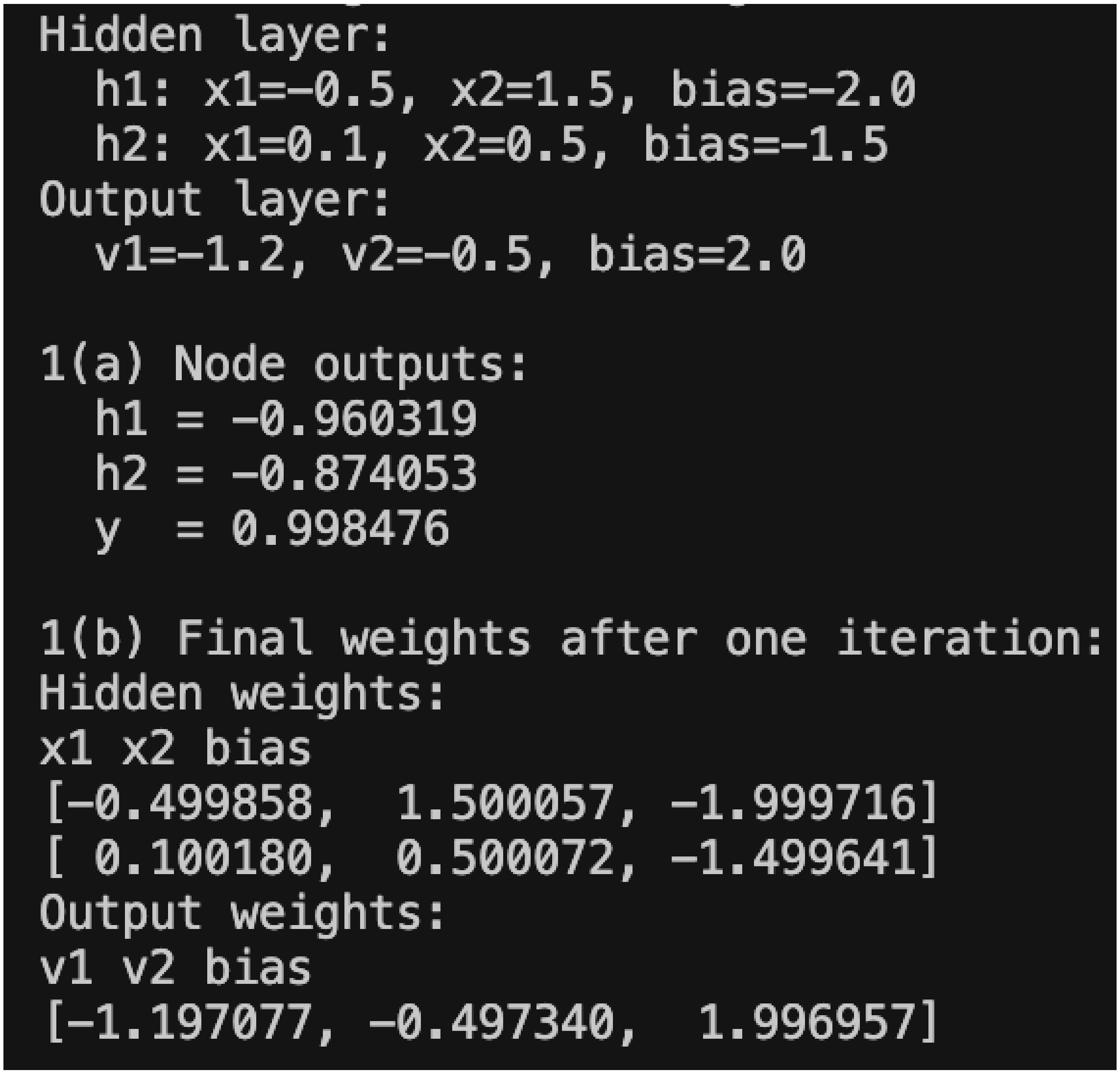
This is Correct!! Fully matches expected output from assignment!

Output for Experiment 1 a) and b)

```
Node outputs:
  h1 = -0.960319
  h2 = -0.874053
  y  = 0.998476
```

1(b) Final weights after one iteration:

```
Hidden weights:
x1          x2          bias
[-0.499858,  1.500057, -1.999716]
[ 0.100180,  0.500072, -1.499641]
Output weights:
v1          v2          bias
[-1.197077, -0.497340,  1.996957]
```



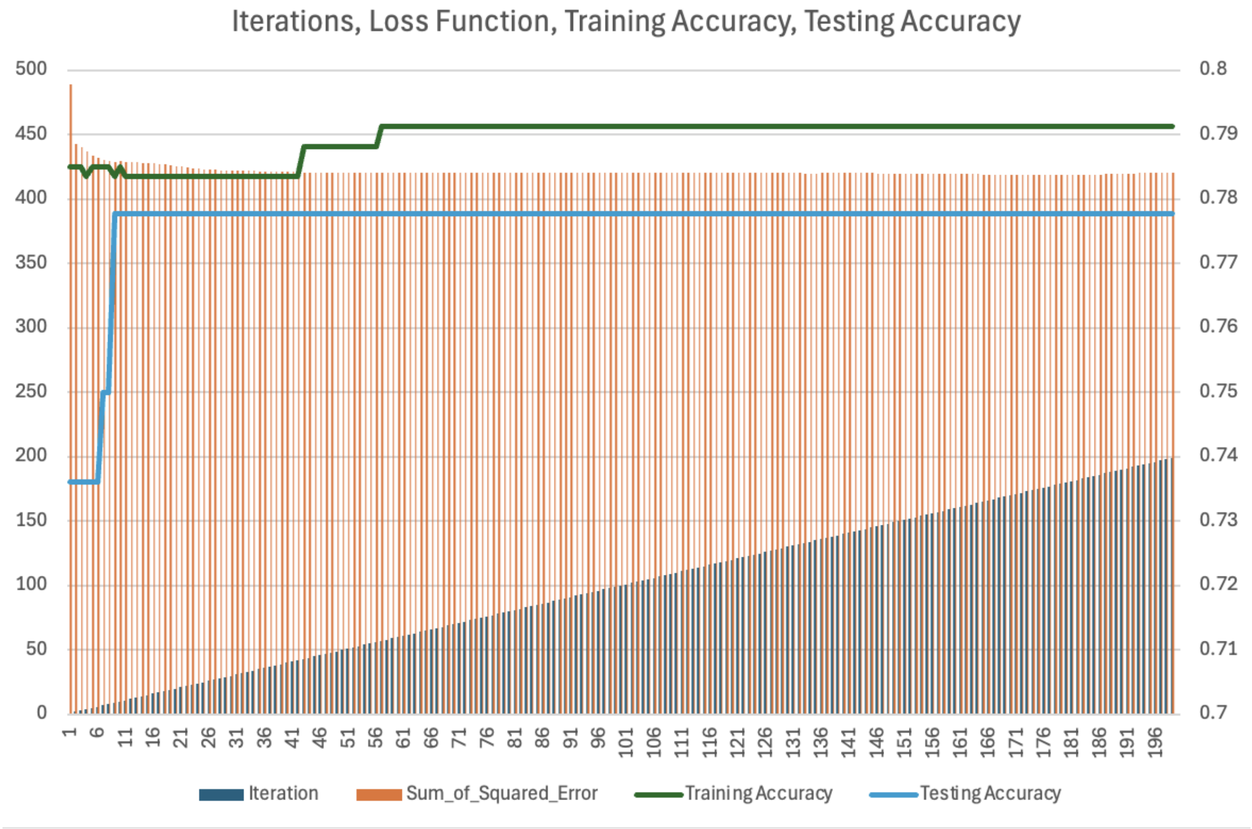
```
Hidden layer:
  h1: x1=-0.5, x2=1.5, bias=-2.0
  h2: x1=0.1, x2=0.5, bias=-1.5
Output layer:
  v1=-1.2, v2=-0.5, bias=2.0

1(a) Node outputs:
  h1 = -0.960319
  h2 = -0.874053
  y  = 0.998476

1(b) Final weights after one iteration:
Hidden weights:
x1 x2 bias
[-0.499858,  1.500057, -1.999716]
[ 0.100180,  0.500072, -1.499641]
Output weights:
v1 v2 bias
[-1.197077, -0.497340,  1.996957]
```

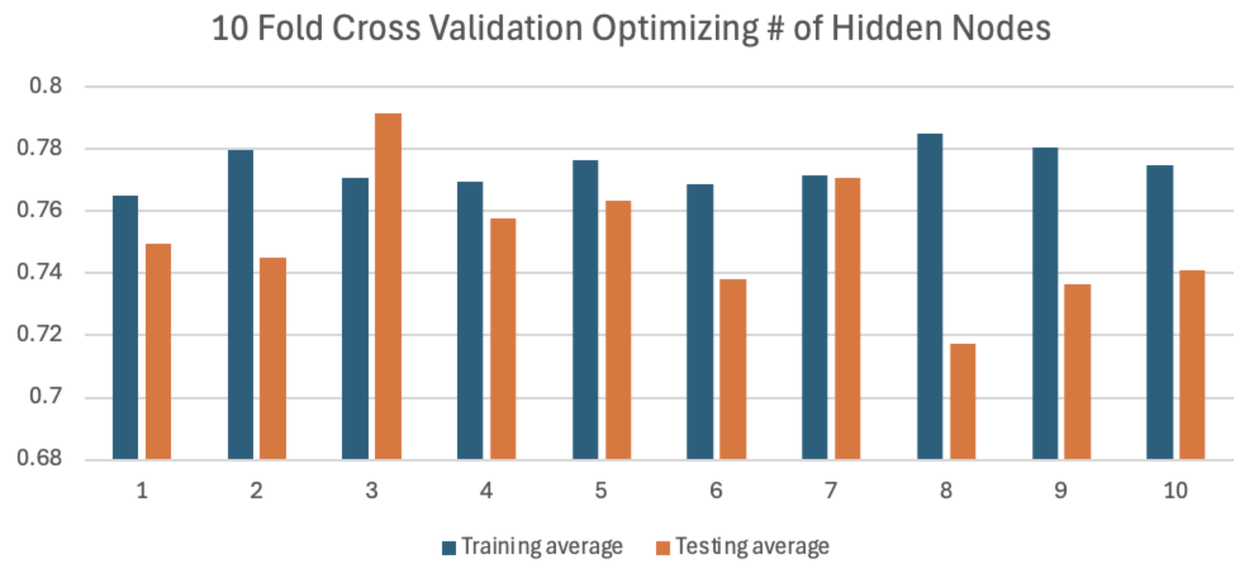
There isn’t much to say about this experiment other than my confidence in the results. I ran with the data from Figure 1 first and got the expected output, so I’m confident in my results.

2)



Graphing the data (it took me 15 minutes to figure out how to have 2 y axes in excel) showed that the sum of squared error loss decreases sharply and then levels off, indicating that the network quickly learned the basic patterns in the data before reaching a plateau. Both training and testing accuracy rose in near step functions and then stabilized around 78–79%, suggesting that the model had converged. Since the testing accuracy tracks the training accuracy closely and does not decline while the training accuracy continues to rise, there is no evidence of overfitting. The training and testing performance were nearly identical especially compared to previous assignments, implying that the model generalizes reasonably well to unseen examples.

3)



The 10-fold cross-validation results show that network size has a noticeable impact on performance. Training accuracy remains relatively stable across all configurations (76.5%-78.5%), while testing accuracy varies more significantly (71.5%-79.0%).
The best performing model uses 3 hidden nodes, achieving the highest test accuracy of 79.0% with a training accuracy of 77.0%. This configuration demonstrates excellent generalization, with test performance actually exceeding training performance. Interestingly, middle sized networks (3, 5, 7 hidden nodes) had the best performance, while the worst performers were 8 hidden nodes (71.5% test accuracy) and 6 hidden nodes (73.7%). This relationship suggests that the optimal network capacity for this dataset falls in the 3-5 hidden node range, where the model has sufficient depth without overfitting.

Overall, I would use 3 hidden nodes as it achieved the best test accuracy while maintaining good generalization (no significant gap between training and test performance).
4)

Hidden Nodes	Learning Rate	Training average	Testing average
1	0.01	0.777005	0.790235
1	0.05	0.777304	0.76169
1	0.1	0.772649	0.750648
1	0.15	0.779019	0.759174
1	0.2	0.774963	0.778592
2	0.01	0.784155	0.745315

2	0.05	0.778249	0.78415
2	0.1	0.770314	0.740563
2	0.15	0.769075	0.756958
2	0.2	0.742206	0.730573
3	0.01	0.790538	0.744939
3	0.05	0.779496	0.740714
3	0.1	0.761285	0.759981
3	0.15	0.760833	0.763399
3	0.2	0.765139	0.763474
4	0.01	0.789294	0.76169
4	0.05	0.780581	0.764808
4	0.1	0.774816	0.733521
4	0.15	0.768286	0.73138
4	0.2	0.754612	0.764657
5	0.01	0.78883	0.784451
5	0.05	0.777321	0.724038
5	0.1	0.772803	0.740563
5	0.15	0.771548	0.73093
5	0.2	0.756153	0.751756

In experiment 2, I observed that the training and testing accuracies flatlined after about the 60th iteration, so I decided to experiment with only the hidden nodes and learning rate. For consistency purposes, I set the iterations at 200 and tested a combination of learning rate {0.01, 0.05, 0.10, 0.15, 0.20} and hidden notes {1, 2, 3, 4, 5} to find the optimal model parameters. Each of the 25 parameters combinations was evaluated using 10-fold cross-validation on the Titanic training dataset, resulting in 250 total model trainings (see the results in the table above).

The optimal configuration achieved a 10-fold cross-validation accuracy of 79.02% with the following parameters:

- Hidden nodes: 1
- Learning rate (η): 0.01
- Iterations: 200

Interestingly, this model also showed excellent generalization with a training accuracy of only 77.70%, which is surprisingly lower than the test accuracy—a positive indicator that the model is not overfitting. I had the following observations running this experiment:

- **Hidden Nodes:** Contrary to my expectations, the simpler networks (1-2 hidden nodes) generally outperformed more complex ones. The single hidden node model achieved the best test accuracy, while models with 3-5 hidden nodes showed more variable performance and signs of overfitting (higher training accuracy than test accuracy).
- **Learning Rate:** Lower learning rates (η = 0.01, 0.05) consistently produced better results than higher rates. Models with η = 0.20 showed worse performance across all hidden node configurations, likely due to overshooting optimal weight values during gradient descent. The optimal learning rate of 0.01 suggests that slower convergence is beneficial for this dataset.
- **Varied Performance:** Test accuracies ranged from 72.4% to 79.0%, demonstrating that parameter selection significantly impacts model performance—a 6.6 percentage point difference between best and worst configurations.