Student Number: <u>20053722</u>   Name: <u>Bryan Hoang</u>

,,,

1. (2 marks) Build a (supervised) neural network to classify the wines (Supervised neural network = multilayer perceptron = MLP in the jargon.)

   Because of the time required, use only 5-fold cross validation. Report briefly on the performance of this technique, including computational time required.

   Neural networks often perform better when their inputs are normalized, often to the interval [0,1] or [-1,+1]. Try using the Normalizer node to see if this helps.

   Answer:

Unnormalized data

A Supervised Neural Network (SNN) with 4 nodes in the hidden layer ($\sqrt{13 \text{ attributes}} \approx 4$) was built, as seen in Figure **??**.
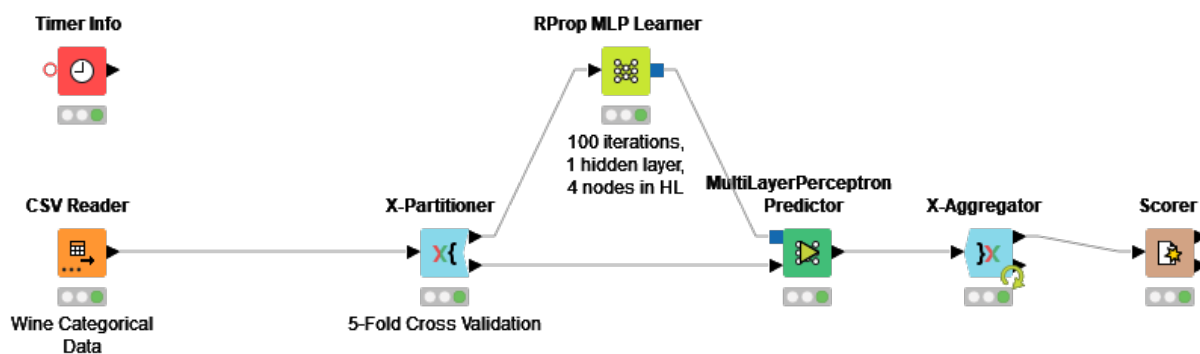

Figure 1: KNIME workflow of a SNN for unnormalized data

Based on the confusion matrix for the SNN shown in Figure **??**, the accuracy of $\sim$47 % leaves a lot of room for improvement.

Student Number: <u>20053722</u> Name: <u>Bryan Hoang</u>

Kind of win...	Type1	Type2	Type3
Type1	12	47	0
Type2	0	71	0
Type3	0	48	0

Correct classified: 83   Wrong classified: 95

Accuracy: 46.629 %   Error: 53.371 %

Cohen's kappa (κ) 0.119

Figure 2: Confusion matrix of the SNN with unnormalized data

Timing wise, the SNN with unnormalized data took ~30 ms to compute, as seen in Figure **??**.

S	Name	L	Execution Time
	RProp MLP Learner		18
	MultiLayerPerceptron Predictor		8
	X-Partitioner		3
	X-Aggregator		1

Figure 3: Timer information for SNN with unnormalized data

Normalized data

Normalized to $[0, 1]$

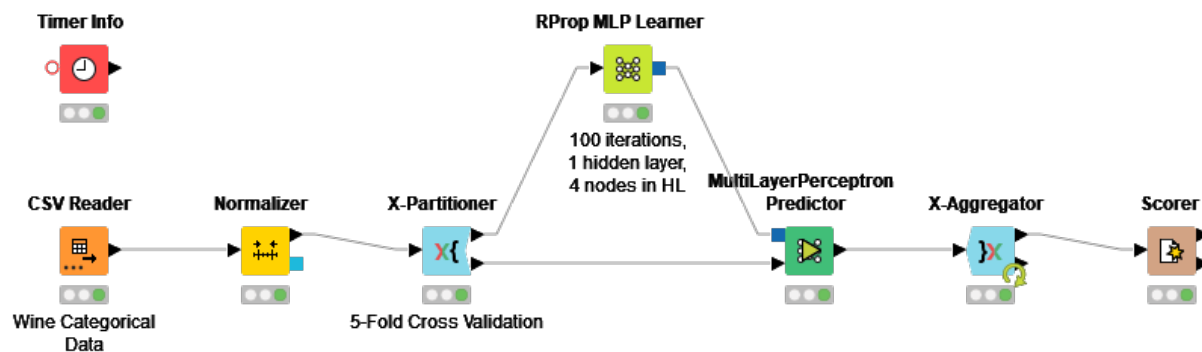I used the Normalizer node to normalize the data to the interval [0,1].

Figure 4: KNIME workflow of a SNN for normalized data

From the confusion matrix for the SNN shown in Figure **??**, the accuracy of $\sim$96 % is a great 39 % increase compared to using unnormalized data.

Kind of win...	Type1	Type2	Type3
Type1	58	1	0
Type2	1	65	5
Type3	0	0	48

Correct classified: 171   Wrong classified: 7

Accuracy: 96.067 %   Error: 3.933 %

Cohen's kappa (κ) 0.941

Figure 5: Confusion matrix of the SNN with data normalized to $[0, 1]$

From Figure **??**, it can be seen that the SNN took $\sim$42 ms to compute, which seems like an acceptable increase in execution time for the significant increase in prediction accuracy.

Student Number: <u>20053722</u>   Name: <u>Bryan Hoang</u>

S	Name	L	Execution Time
	RProp MLP Learner		20
	MultiLayerPerceptron Predictor		11
	X-Partitioner		2
	X-Aggregator		1
	Normalizer		8

Figure 6: Timer information for SNN with data normalized to $[0, 1]$

Student Number: <u>20053722</u> Name: <u>Bryan Hoang</u>

Normalized to $[-1, 1]$

From the confusion matrix for the SNN shown in Figure **??**, the accuracy of ∼98 % is even better than the ∼96 % accuracy obtained from normalizing to the interval $[0, 1]$.

Kind of win...	Type1	Type2	Type3
Type1	59	0	0
Type2	2	68	1
Type3	0	0	48

Correct classified: 175   Wrong classified: 3

Accuracy: 98.315 %   Error: 1.685 %

Cohen's kappa (κ) 0.974

Figure 7: Confusion matrix of the SNN with data normalized to $[-1, 1]$

From Figure **??**, it can be seen that the SNN took ∼44 ms to compute, which also seems like an acceptable increase in execution time for the significant increase in prediction accuracy.

S Name	L Execution Time
RProp MLP Learner	18
MultiLayerPerceptron Predictor	9
X-Partitioner	2
X-Aggregator	2
Normalizer	11

Figure 8: Timer information for SNN with data normalized to $[-1, 1]$