For a large amount of data, I prefer to use SPSS because it specializes in statistical analysis. So, I will list the results of the SPSS analysis below.

1. Preliminary Analysis of training data

Before running the neural network, the first thing to do is to make a preliminary analysis of the data. So, I run dispersion analysis and posterior dispersion analysis, including Maximum, Minimum, Mean, Std. Deviation, Variance, Skewness, and Kurtosis. The results are shown below

```
FREQUENCIES VARIABLES=x1_train x2_train x3_train y_train
/STATISTICS=STDDEV VARIANCE MINIMUM MAXIMUM SEMEAN MEAN SKEWNESS SESKEW KURTOSIS SEKURT
/ORDER=ANALYSIS.
```

Statistics

		x1_train	x2_train	x3_train	y_train
N	Valid	10000	10000	10000	10000
	Missing	0	0	0	0
Mean		.4951931712	.5040919527	.4975040397	.4474590381
Std. Error of Mean		.0028900413	.0028819643	.0028640249	.0018593230
Std. Deviation		.2890041270	.2881964308	.2864024892	.1859322962
Variance		.084	.083	.082	.035
Skewness		-.004	-.024	.008	.048
Std. Error of Skewness		.024	.024	.024	.024
Kurtosis		-1.199	-1.192	-1.173	-.580
Std. Error of Kurtosis		.049	.049	.049	.049
Minimum		.0003289775	.0002434520	.0000554176	.0088996756
Maximum		.9999421806	.9999759923	.9999548955	.9889424485

Figure 1

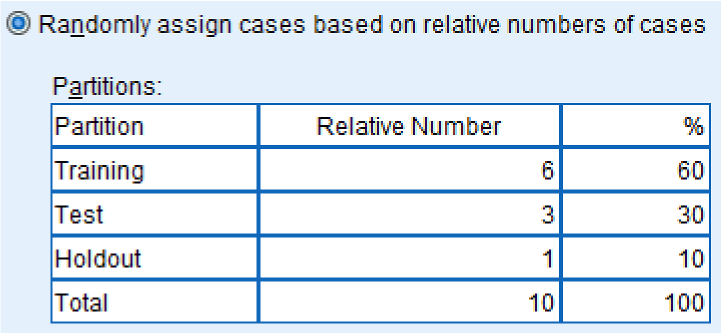
For detail, please kindly find the attached file " 1. Preliminary Analysis.spv " (Open with SPSS). Meanwhile, I already export this file from SPSS. Please kindly find attached file " 1. Preliminary Analysis.pdf ".

The results showed that the dataset is no missing data, and the data range is between 0 and 1. Moreover, the Skewness and Kurtosis showed that the data is close to normal distribution.

In my opinion, I think the raw data doesn't need to standardize or normalize. As for the choice of activation function, according to the data range, the Sigmoid Function is the best choice for hidden layers activation function. Meanwhile, the Identity Function is the best choice for the output layer activation function, from my perspective.

2. Construct Multi-Layer Perceptron (MLP) Model

Many people divide 70% of the dataset into training samples and 30% of the dataset into testing samples but ignore the holdout samples. From my perspective, I decided to partition 60% of the dataset into training samples, 30% of the dataset into testing samples, and 10% of the dataset into holdout samples.


Randomly assign cases based on relative numbers of cases

Partitions:

Partition	Relative Number	%
Training	6	60
Test	3	30
Holdout	1	10
Total	10	100

Figure 2

According to the preliminary analysis of the dataset and question requirements, I set two hidden layers and four units in each layer. As mentioned above, I choose the Sigmoid Function as hidden layers activation function, and the Identity Function with non rescale as output layer activation function.

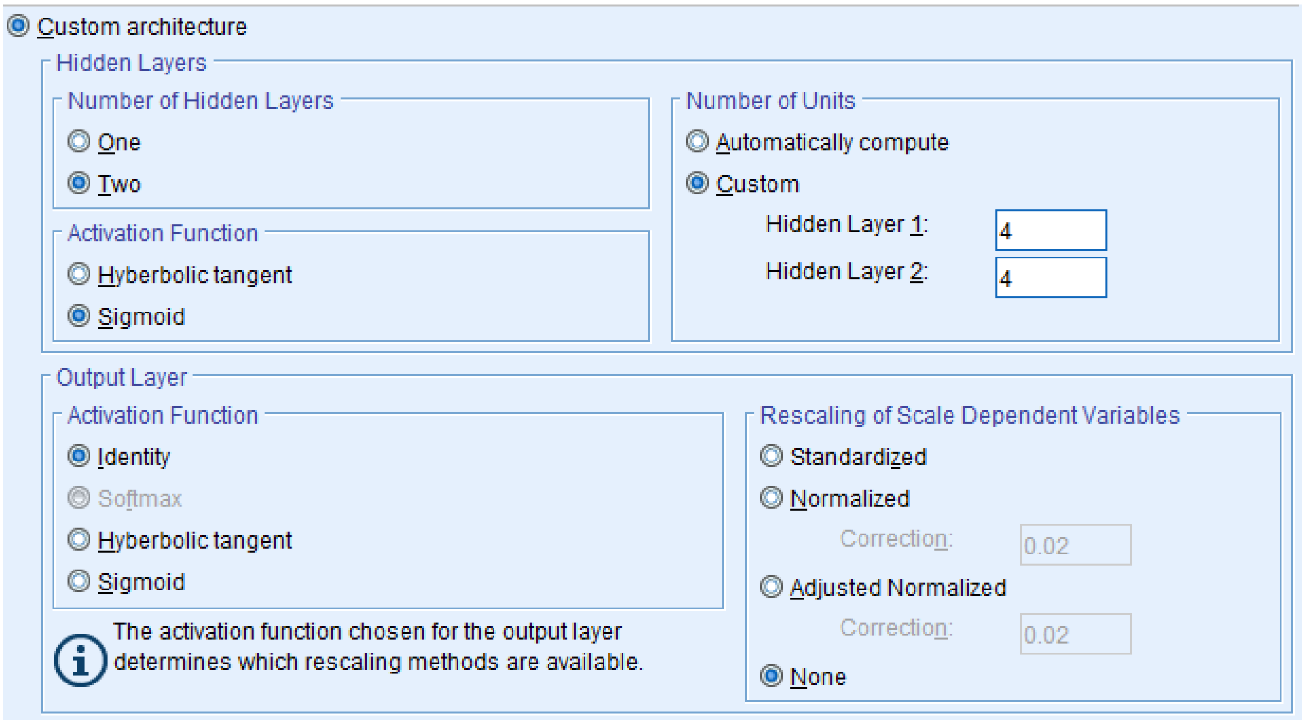


Figure 3

As for the training type, I prefer the Batch type, which updates the synaptic weights only after passing all training data records. Also, the Batch type is the only type to correspond to the scaled conjugate gradient, which can directly minimize the total error in these two optimization algorithms.

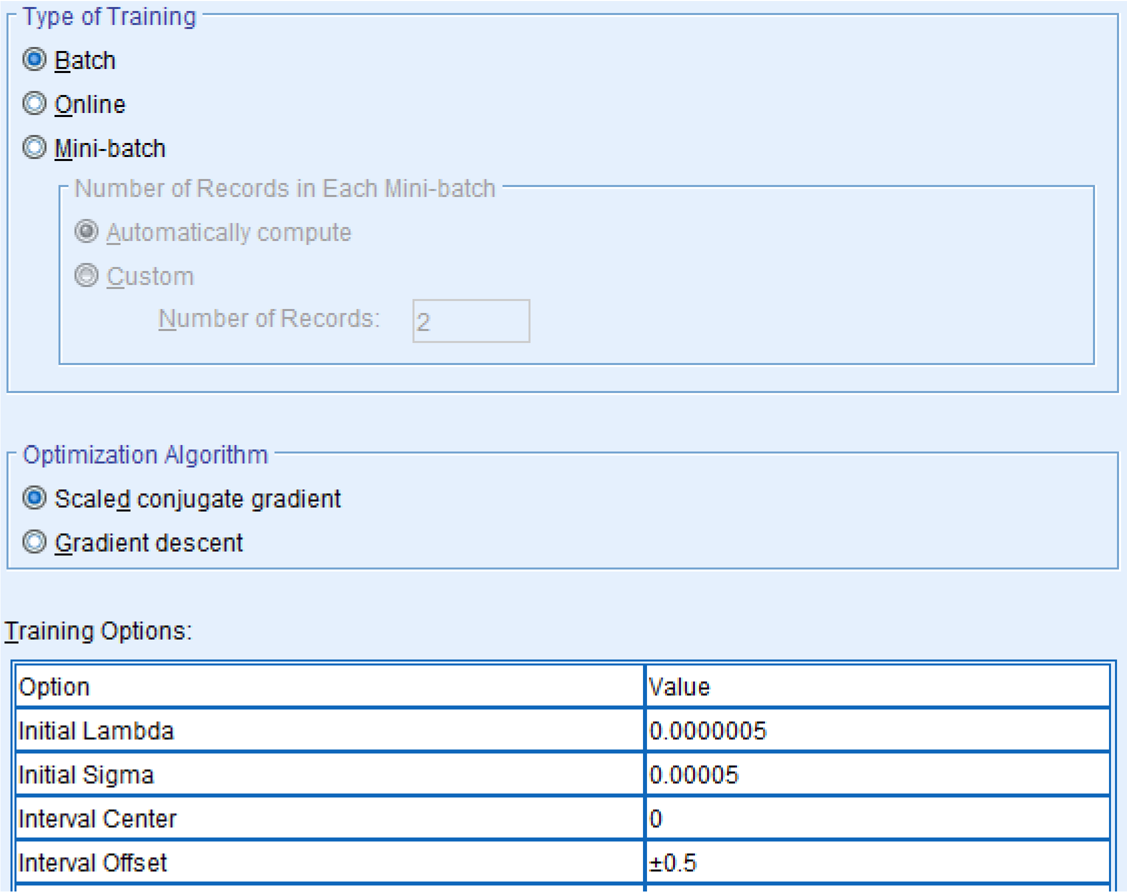


Figure 4

Finally, the specific Multi-Layer Perceptron (MLP) model results are shown below

<u>For detail, please kindly find the attached file " 2. Multi-Layer Perceptron (MLP) Model.spv " (Open with SPSS). Meanwhile, I already export this file from SPSS. Please kindly find attached the file " 2. Multi-Layer Perceptron (MLP) Model.pdf ".</u>

```
*Multilayer Perceptron Network.
MLP y_train (MLEVEL=S) WITH x1_train x2_train x3_train
 /RESCALE COVARIATE=NONE DEPENDENT=NONE
  /PARTITION  TRAINING=6  TESTING=3  HOLDOUT=1
  /ARCHITECTURE   AUTOMATIC=NO HIDDENLAYERS=2 (NUMUNITS=4,4) HIDDENFUNCTION=SIGMOID
    OUTPUTFUNCTION=IDENTITY
  /CRITERIA TRAINING=BATCH OPTIMIZATION=SCALEDCONJUGATE LAMBDAINITIAL=0.0000005
    SIGMAINITIAL=0.00005 INTERVALCENTER=0 INTERVALOFFSET=0.5 MEMSIZE=1000
  /PRINT CPS NETWORKINFO SUMMARY SOLUTION IMPORTANCE
  /PLOT NETWORK PREDICTED RESIDUAL
  /SAVE PREDVAL(y_predict )
  /STOPPINGRULES ERRORSTEPS= 1 (DATA=AUTO) TRAININGTIMER=ON (MAXTIME=15) MAXEPOCHS=AUTO
    ERRORCHANGE=1.0E-4 ERRORRATIO=0.001
 /MISSING USERMISSING=EXCLUDE .
```

Figure 5

Case Processing Summary

		N	Percent
Sample	Training	5961	59.6%
	Testing	3018	30.2%
	Holdout	1021	10.2%
Valid		10000	100.0%
Excluded		0	
Total		10000	

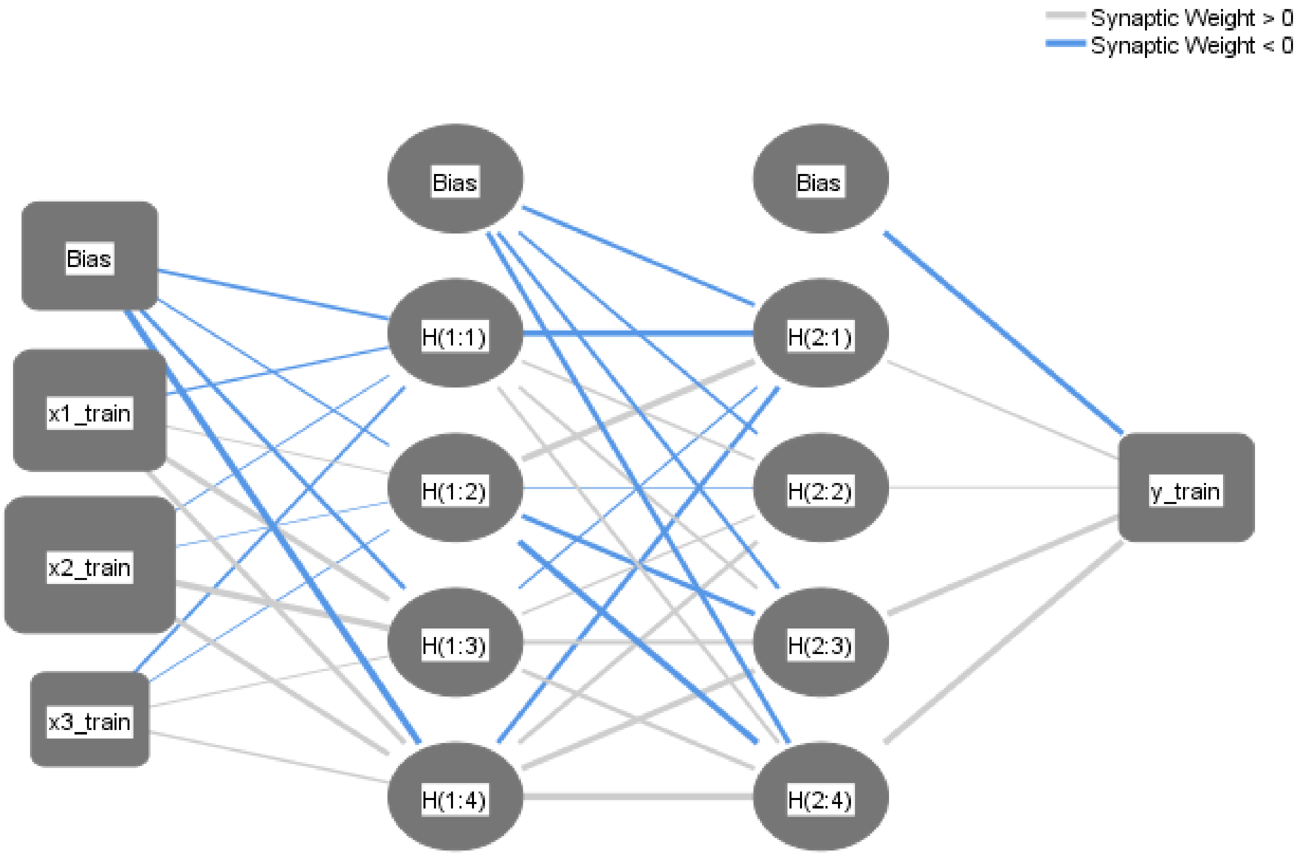
Figure 6

Network Information

Input Layer	Covariates	1	x1_train
		2	x2_train
		3	x3_train
	Number of Units[a]		3
	Rescaling Method for Covariates		None
Hidden Layer(s)	Number of Hidden Layers		2
	Number of Units in Hidden Layer 1[a]		4
	Number of Units in Hidden Layer 2[a]		4
	Activation Function		Sigmoid
Output Layer	Dependent Variables	1	y_train
	Number of Units		1
	Rescaling Method for Scale Dependents		None
	Activation Function		Identity
	Error Function		Sum of Squares

a. Excluding the bias unit

Figure 7



Hidden layer activation function: Sigmoid

Output layer activation function: Identity

Figure 8

Model Summary

Training	Sum of Squares Error	1.391
	Relative Error	.014
	Stopping Rule Used	1 consecutive step(s) with no decrease in error[a]
	Training Time	0:00:00.08
Testing	Sum of Squares Error	.683
	Relative Error	.013
Holdout	Relative Error	.012

Dependent Variable: y_train

a. Error computations are based on the testing sample.

Figure 9

Parameter Estimates

Predictor		Predicted								
		Hidden Layer 1				Hidden Layer 2				Output Layer
		H(1:1)	H(1:2)	H(1:3)	H(1:4)	H(2:1)	H(2:2)	H(2:3)	H(2:4)	y_train
Input Layer	(Bias)	-.485	-.275	-.550	-1.991					
	x1_train	-.393	.106	1.411	.989					
	x2_train	-.099	-.040	1.817	1.146					
	x3_train	-.449	-.093	.211	.400					
Hidden Layer 1	(Bias)					-.542	-.460	-.539	-.946	
	H(1:1)					-1.390	.446	.510	.516	
	H(1:2)					1.626	-.091	-.845	-1.619	
	H(1:3)					-.252	.285	.641	.654	
	H(1:4)					-.811	.700	1.452	2.010	
Hidden Layer 2	(Bias)									-1.381
	H(2:1)									.442
	H(2:2)									.182
	H(2:3)									1.525
	H(2:4)									1.793

Figure 10

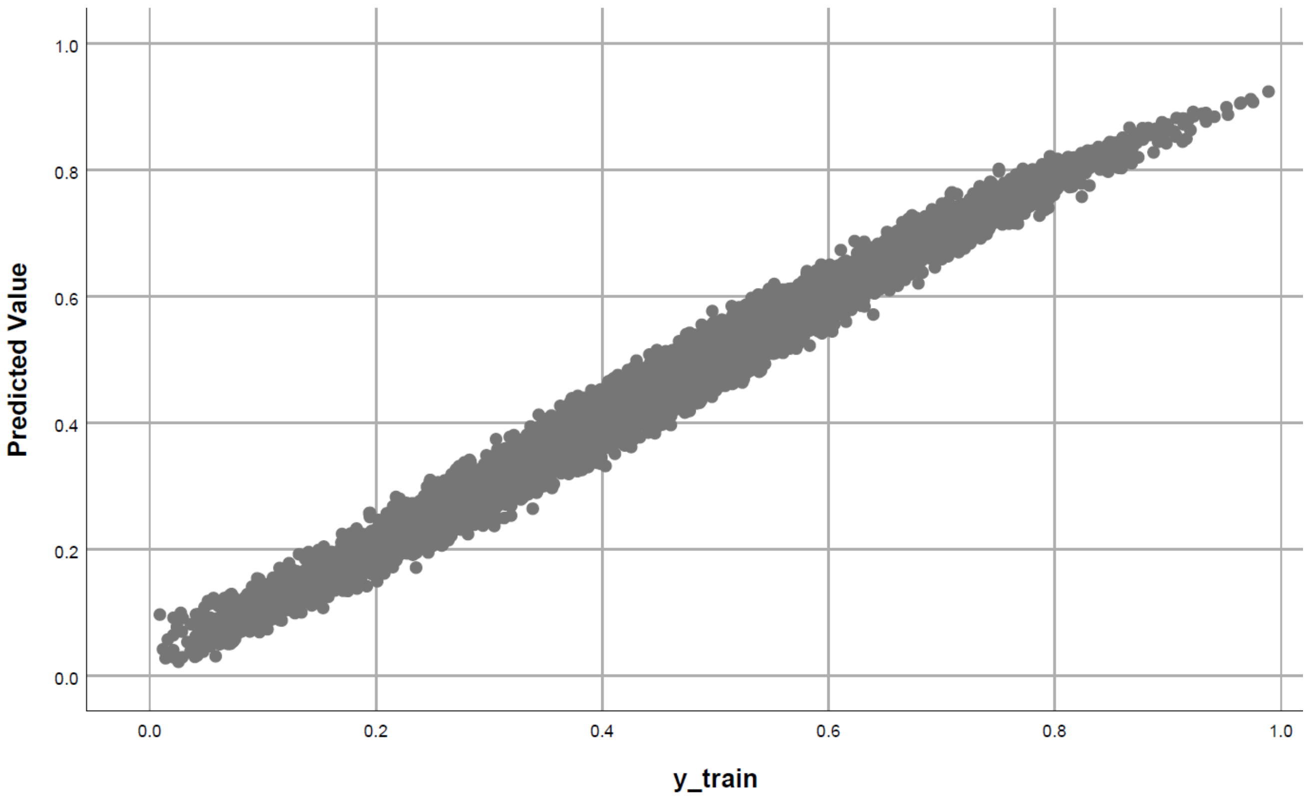


Figure 11

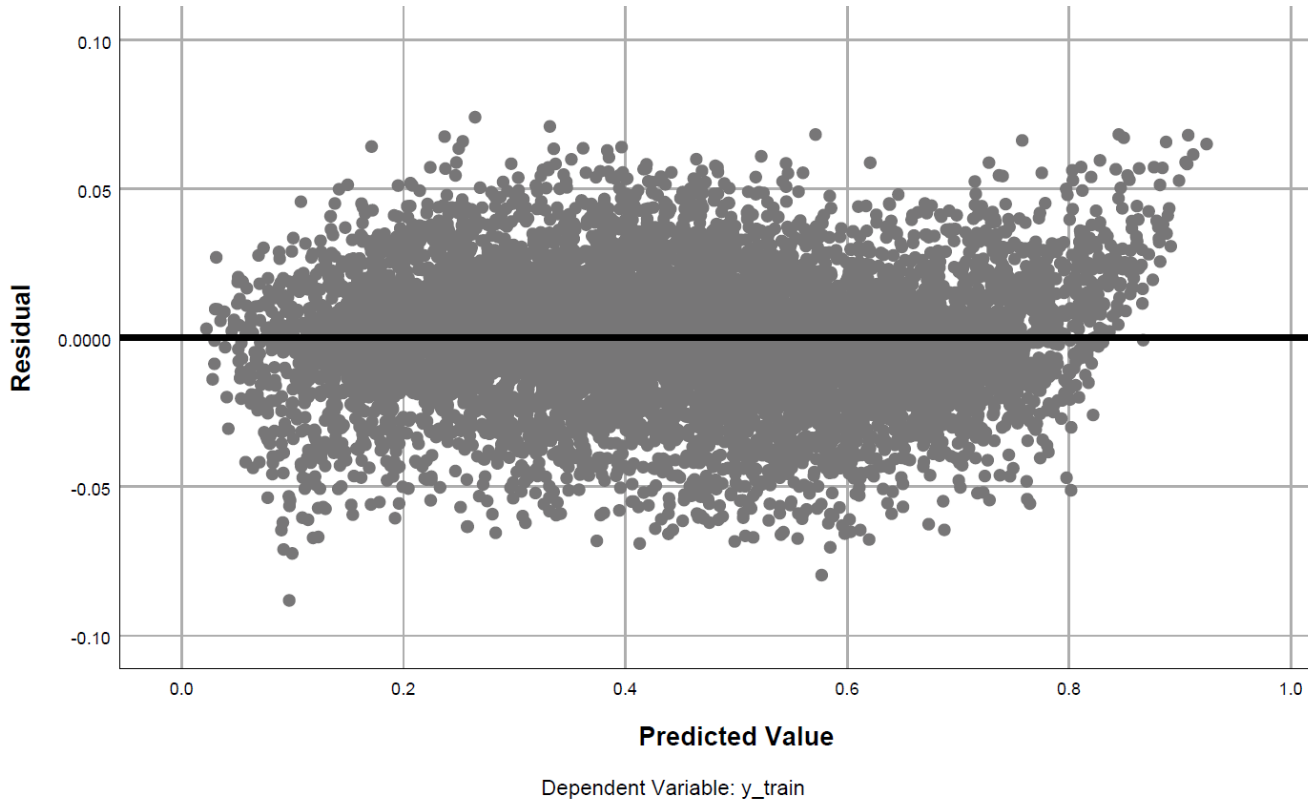


Figure 12

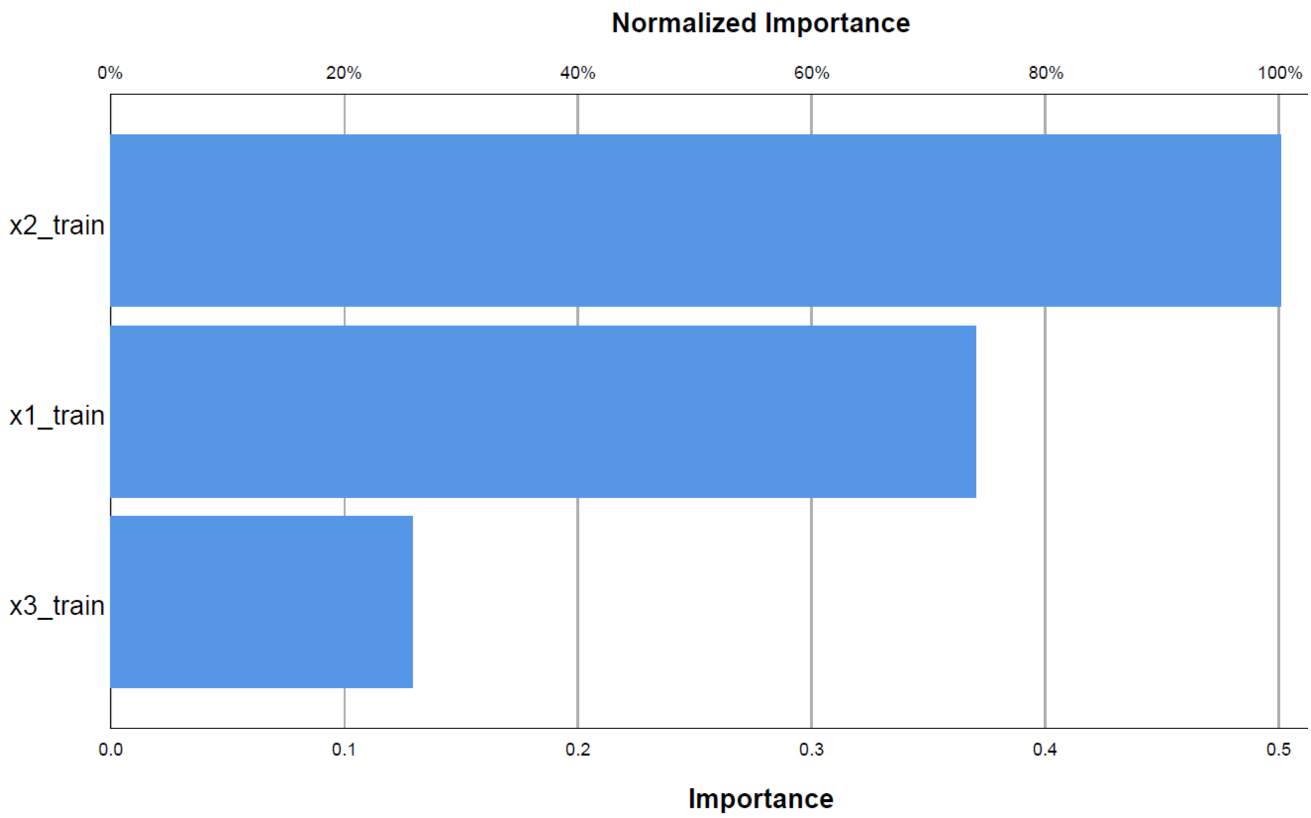


Figure 13

Independent Variable Importance

	Importance	Normalized Importance
x1_train	.370	73.9%
x2_train	.501	100.0%
x3_train	.129	25.8%

Figure 14

The result showed that this Multi-Layer Perceptron (MLP) model has a small Relative Error, which means the model's training is excellent. Also, the predicted value is almost linearly correlated with the y_train value in Figure 12. This relationship post that the prediction of the model is accurate. Moreover, the result illustrated some critical information, such as the precise weight value in every unit (Figure 10) and the importance of these three variables (Figure 13 & 14).

3. Predict the Test Dataset

In SPSS, using Multi-Layer Perceptron (MLP) model to predict is uncomplicated. Just import the test dataset, connect the test dataset down to the training data, and re-do the operation mentioned above.

For the predictive value, please kindly find the attached file " test_predicted.txt ".

4. Weakness

However, one drawback of the SPSS is that there is no way to customize the output. The predictive values can only be kept in the original table automatically. So, The training dataset and the test dataset have to fit together in one table, and I have to extract the predictive values into the text file manually.

For the integrated table, please kindly find the attached file " 3. Integrated Table (Train & Test).sav " (Open with SPSS). Meanwhile, I already export this file from SPSS. Please kindly find the attached file " 3. Integrated Table (Train & Test).xls ".