Neural Networks and Clusterting in Weka

Chulun Zhou

clzhou@link.cuhk.edu.hk

Overview

25-min Neural Network demo
20-min Practice, QA
25-min Clustering Demo & Assignment 2
20-min Practice, QA

- What is Neural Network?
- Building Neural Network with Weka
- Interpreting output (evaluation metrics)
- Visualizing Neural Network using GUI (**graphical user interface**)
- Clustering Algorithms with Weka

What is Neural Network?

Neural Network is a machine learning model inspired by the biological neural networks that simulate what our brain does.

Neural Network has several components including the Input Layer, Hidden Layers and Output Layer:

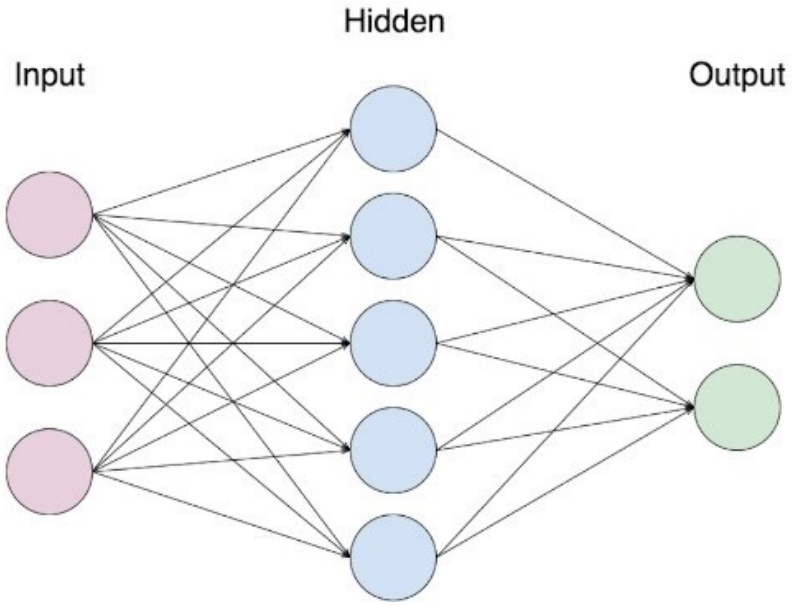
(1) **Input Layer** denotes the input variables that will be fed into the network,

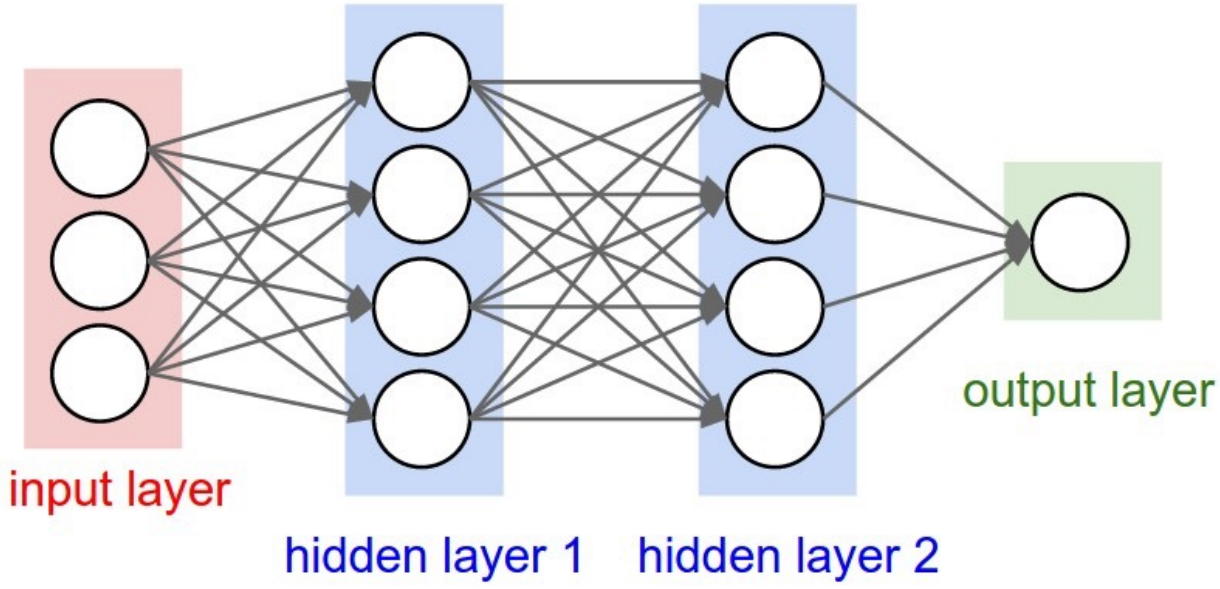
(2) **Hidden Layers** are the computation layers (or parameters) that will be trained,

(3) **Output Layer** denotes the output of the model. For example, the class label in classification task or the real number in regression task.

What is Neural Network?

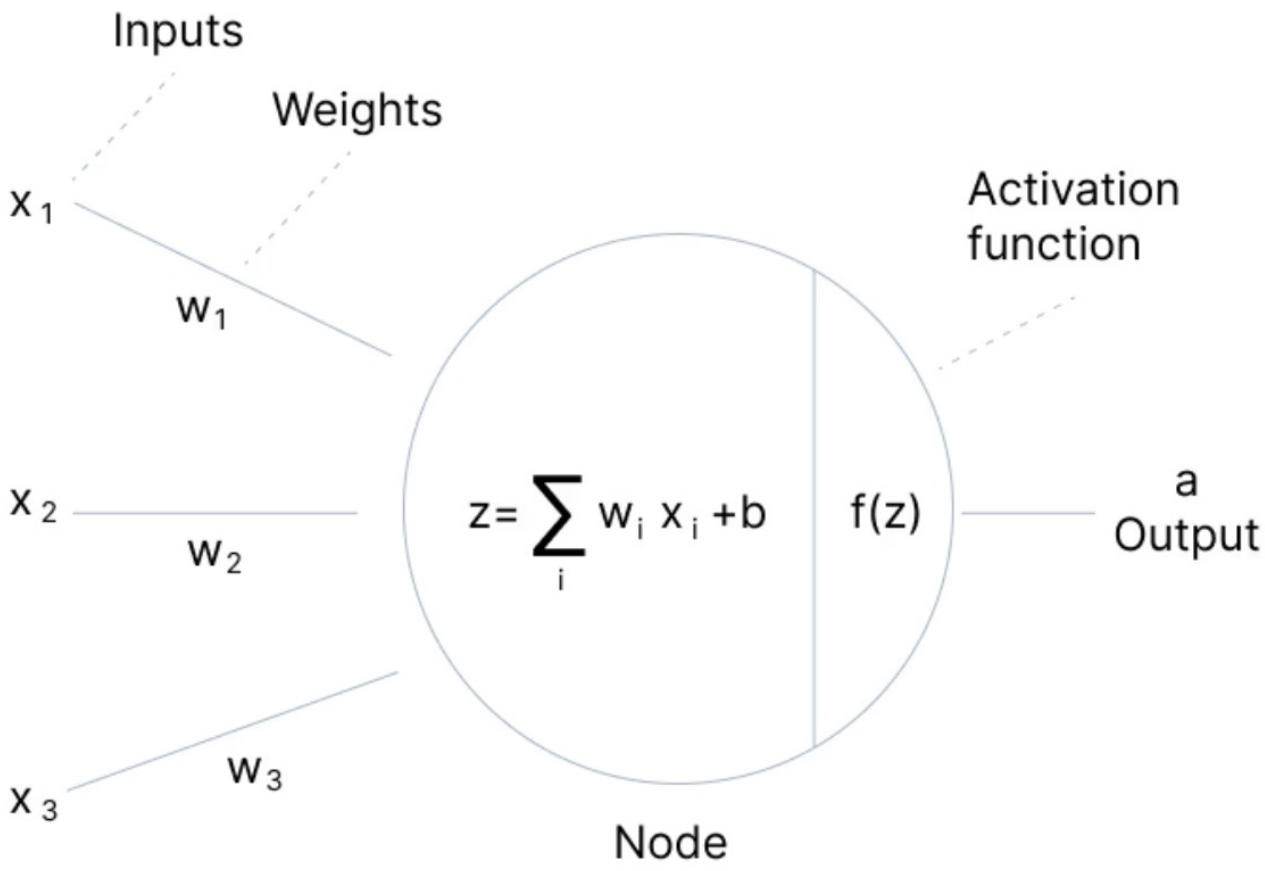
A typical neural network model can be represented as follow:





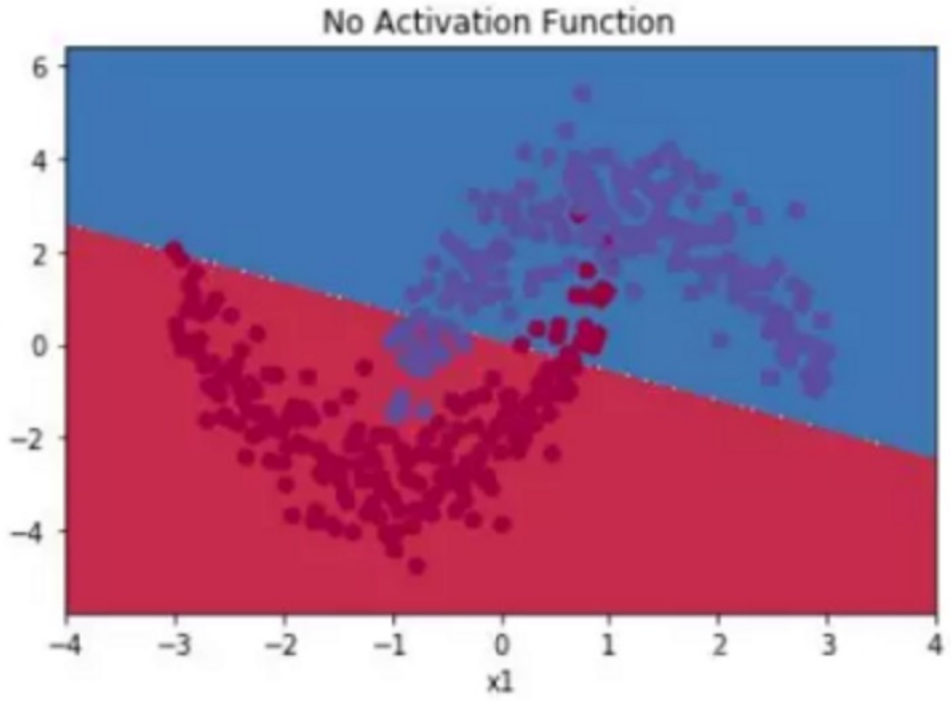
What is Neural Network?

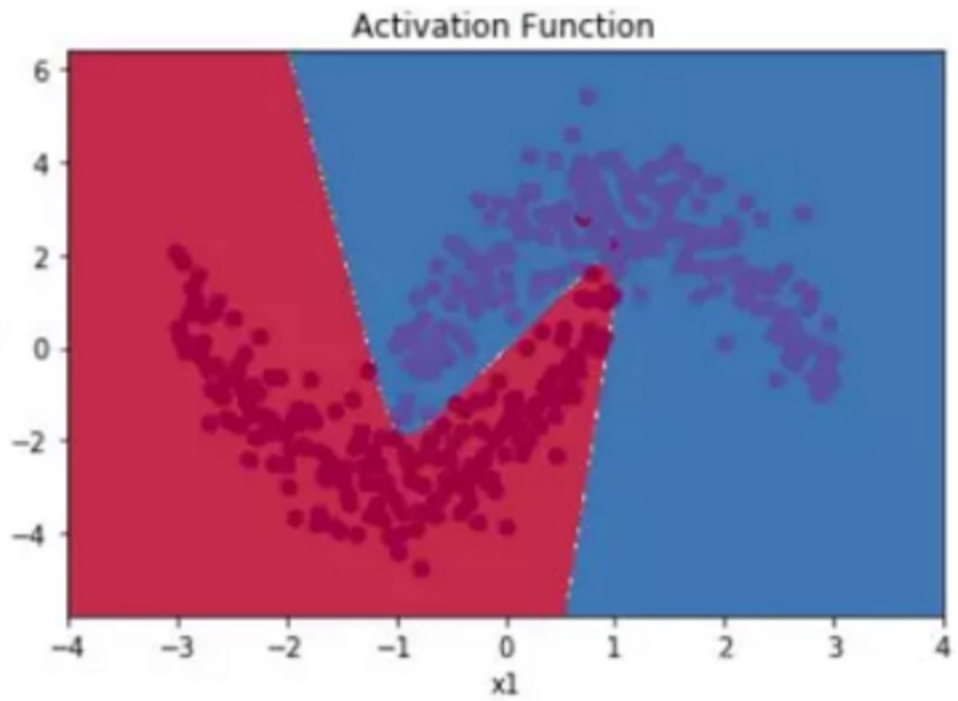
Activation function



What is Neural Network?

Activation function





What is Neural Network?

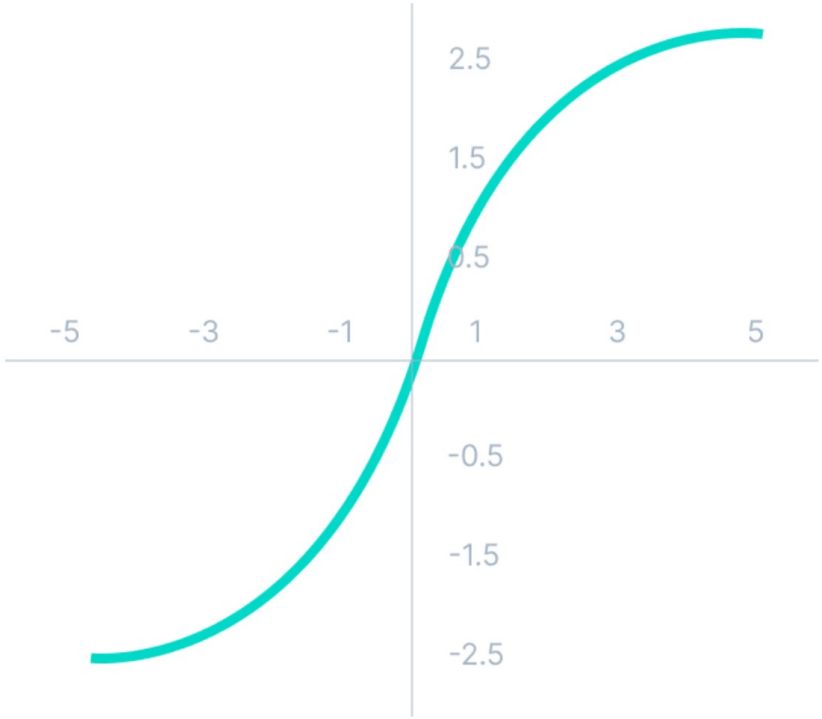
Typical non-linear activation functions

$$f(x) = \frac{1}{1 + e^{-x}}$$

What is Neural Network?

Typical non-linear activation functions

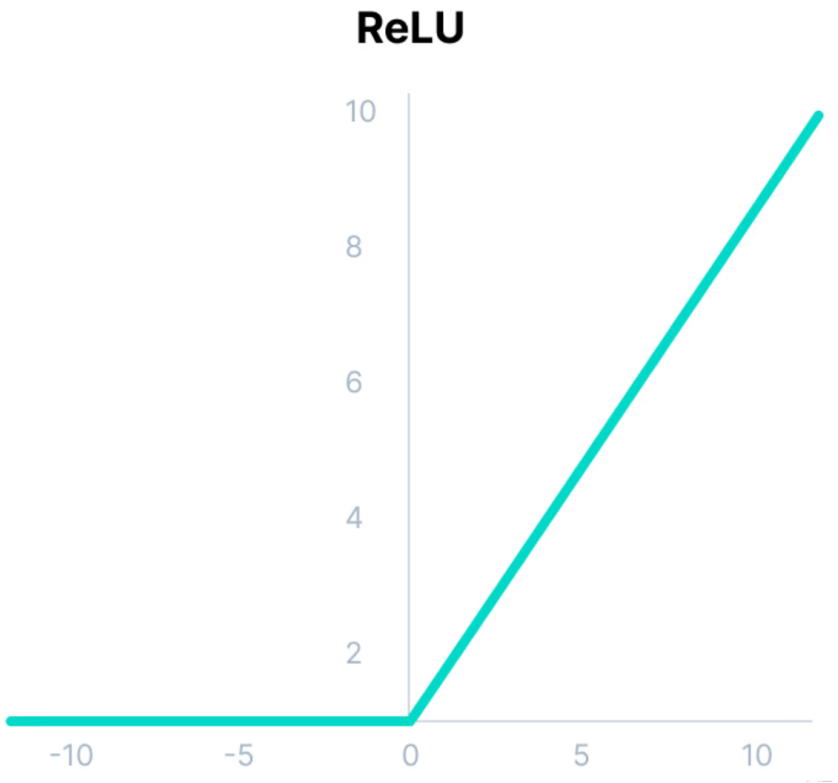
Tanh (Hyperbolic Tangent)



$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

What is Neural Network?

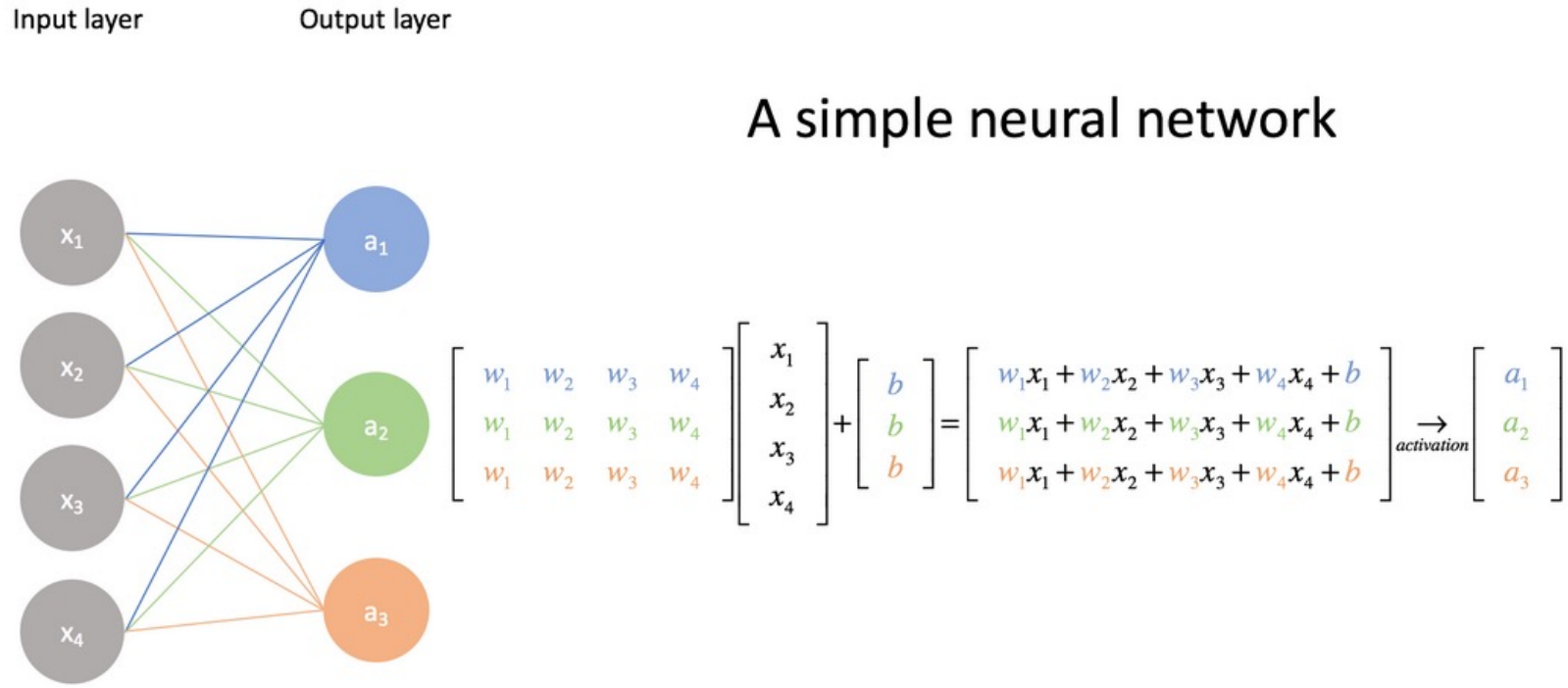
Typical non-linear activation functions



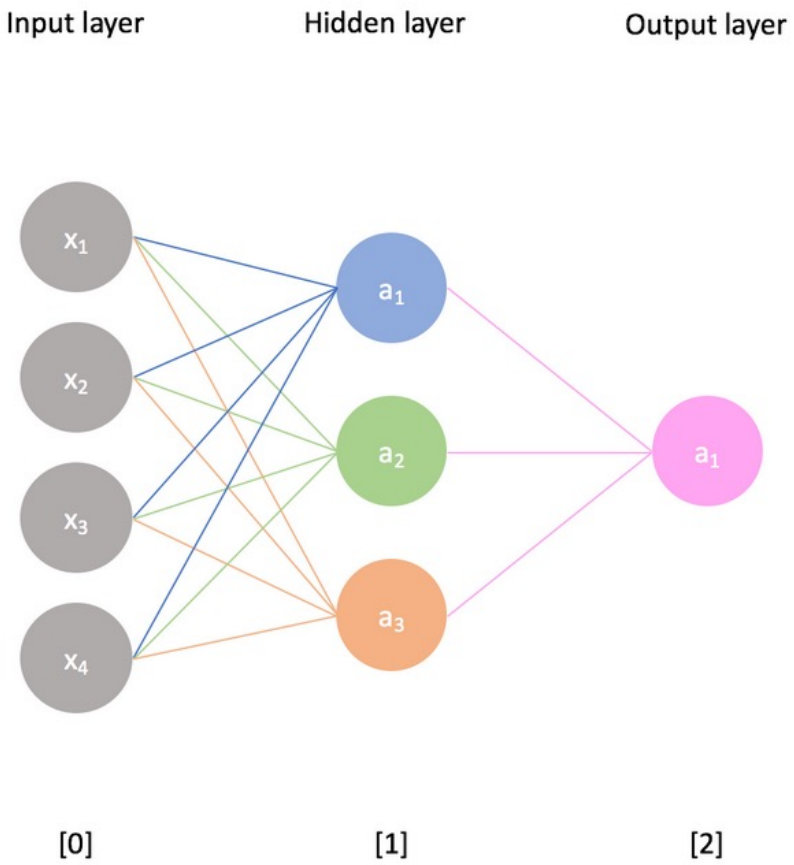
$$f(x) = max\,(0, x)$$

What is Neural Network?

In this figure, the network with this structure is called **Feed-Forward network** or **Multi-layer perceptron**. Neural network is essentially matrix multiplication+activation funtion



What is Neural Network?



I'll use the superscript $[l]$ to refer to the l^{th} layer of the network and the subscript i to refer to the i^{th} neuron in a layer.

$$z_2^{[1]} = w_2^{[1]\mathrm{T}} a^{[0]} + b_2$$

More generally, we can calculate the activation of neuron i in layer l.

$$z_i^{[l]} = w_i^{[l]\mathrm{T}} a^{[l-1]} + b_i^{[l]} \qquad a_i^{[l]} = g\left(z_i^{[l]}\right)$$

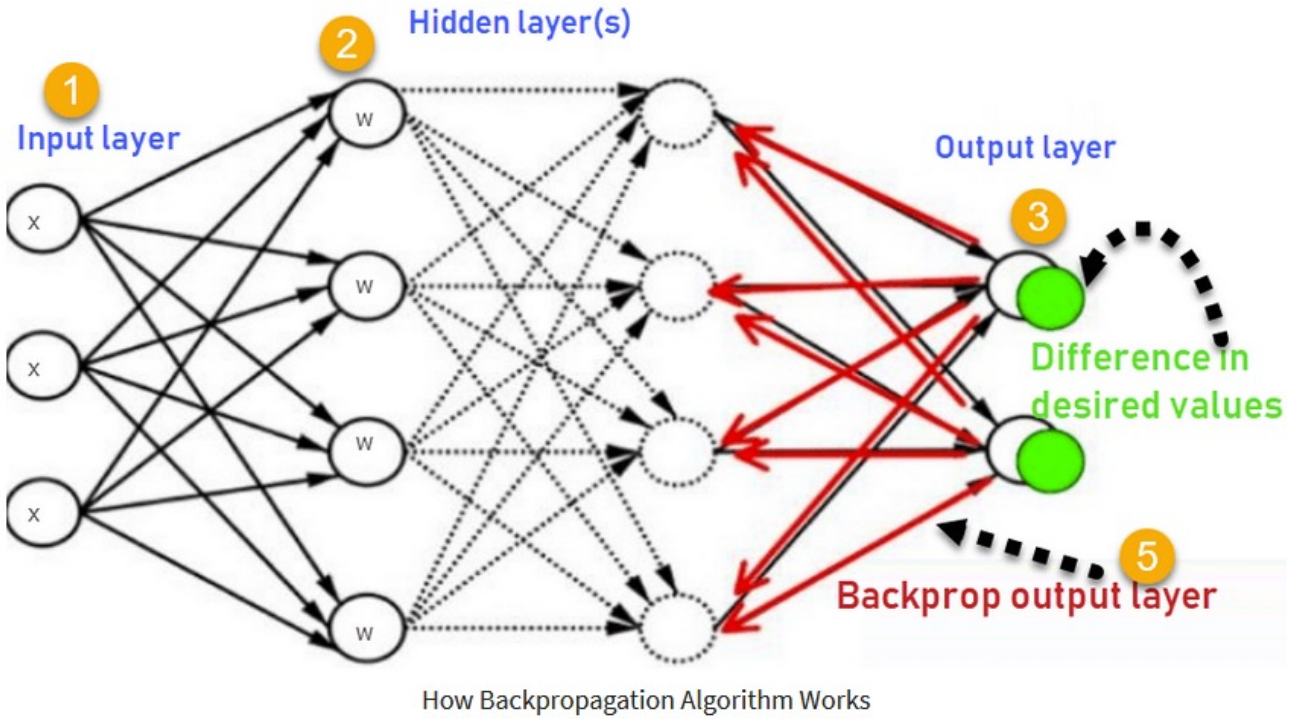
Forward propagation

For $l = 1,2,3 \ldots.$

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g\left(Z^{[l]}\right)$$

What is Neural Network?



How Backpropagation Algorithm Works

Backpropagation: optimizing parameters

Difference in desired values is evaluated using the cost function.

For example, for regression

$$J(W,b,x,y) = \frac{1}{2}||a^L - y||_2^2$$

For classification, cross entropy loss function

$$J(W,b,a,y) = -[ylna + (1-y)ln(1-a)]$$

$$w^k = w^{k-1} - \eta \frac{\partial J}{\partial w^{k-1}}$$

k means the kth interation

What is Neural Network?

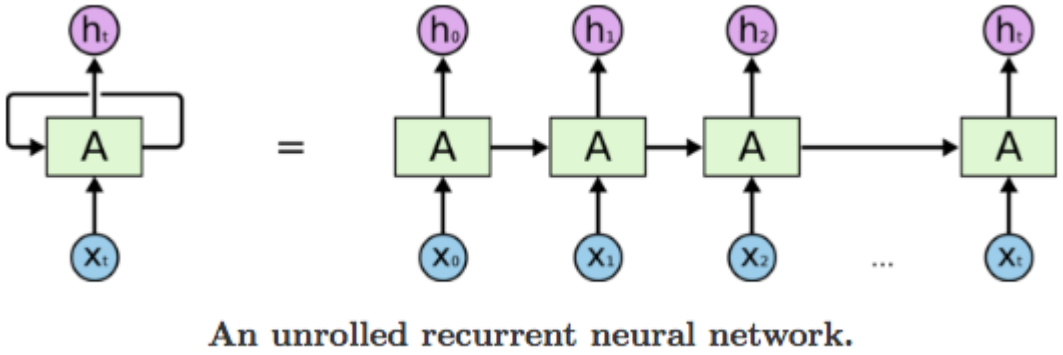
A general training procedure:

```
training data: {(x1,y1),(x2,y2),...}

for i = 1,2,3....n (n is the number of traning epoches):
    for j = 1,2,.....:
        output = Model.forward(xj) #forward propagation
        loss = f(ouput, yj)  # f is a cost function
        loss.backward() #backward propoagration to optimize the parameters in the Model
```

Generally, we input b samples in the forward process. b is also called the batch size.

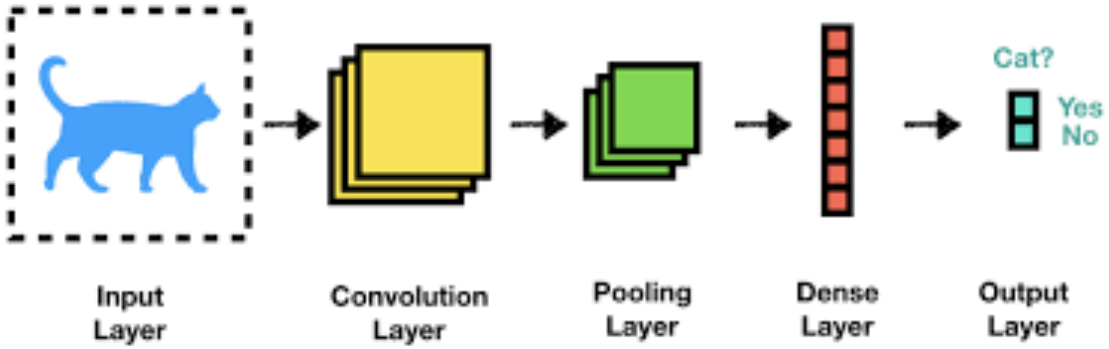
What is Neural Network?

There are some advanced neural networks with special structure. For example, the **recurrent neural network (RNN)** that can be used to handle time-series (weather of each day) or sequential data (text). Or **convolutional neural network (CNN)** that is widely used for handling image.



An unrolled recurrent neural network.

Recurrent Neural Network



Convolutional Neural Network

Neural Network in Weka

In this tutorial, we will focus on building the Multi-layer perceptron using Weka.

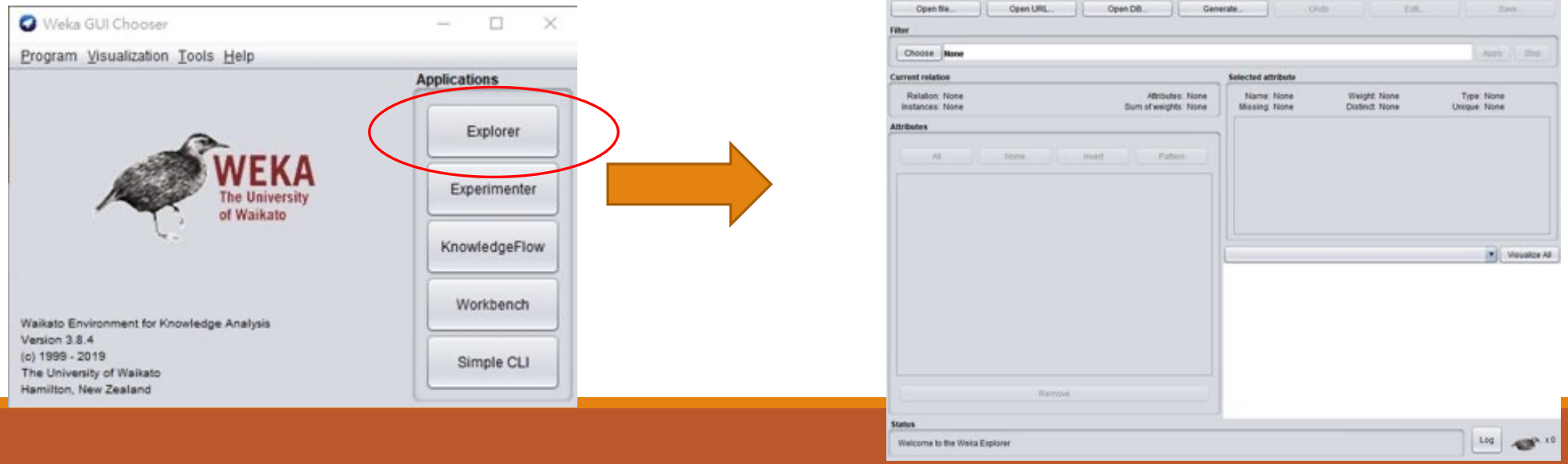
You can use Weka to easily construct a neural network and it will help you to configure most of the setting of it like the activation function and cost function.

All you need is to prepare the data for it.

Preparation for building Neural Network

Before constructing our neural network, again, we first need to prepare our training data.

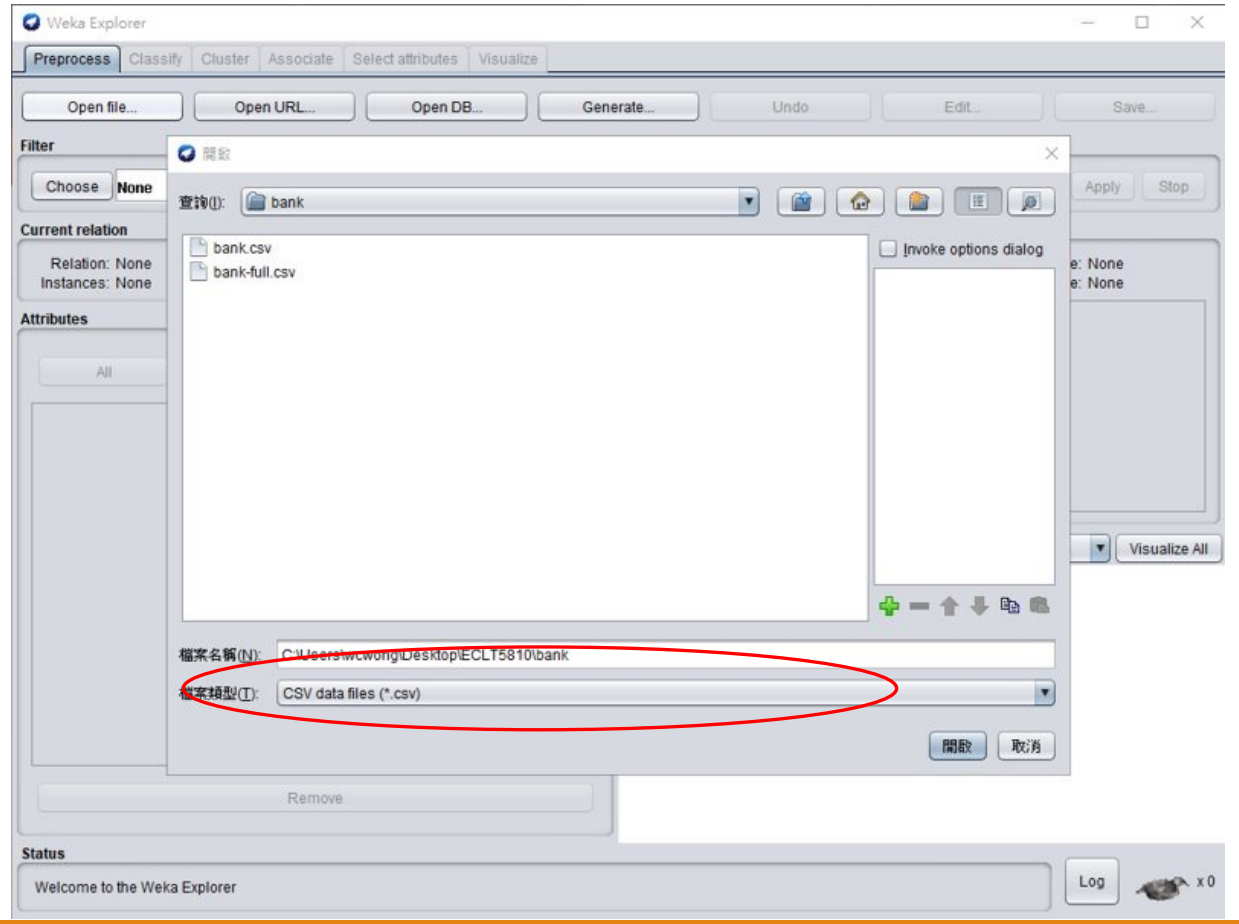
Open Weka, choose **Explorer** in the **Weka GUI Chooser**



Preparation for building Neural Network

Click **Open file**, then open the bank-additional.csv used in the Assignment 1

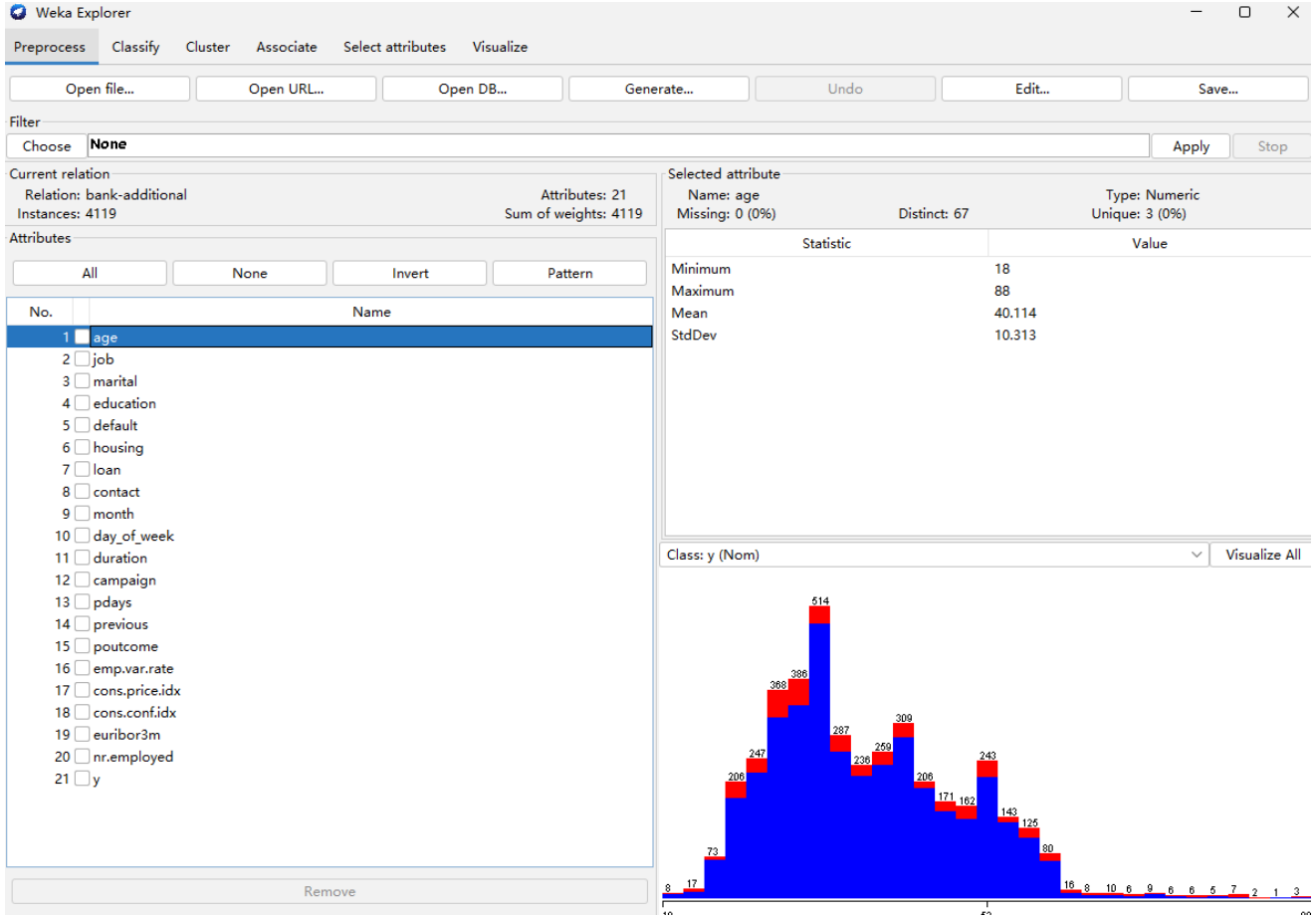
Again, please
remember to change to **CSV data files(*.csv)** in file type.



Preparation for building Neural Network

Now, data is loaded into Explorer.

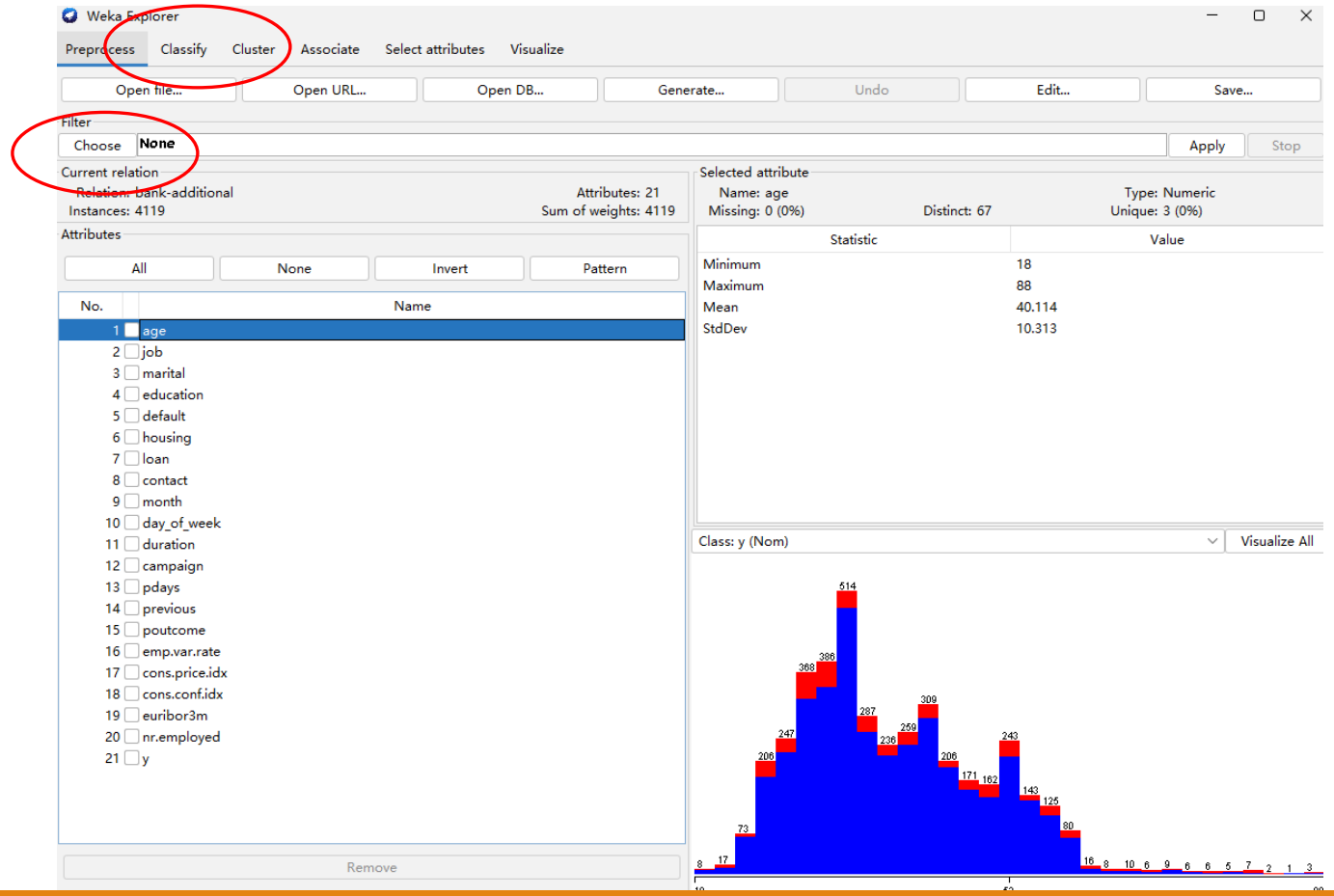
And then we can perform feature engineering before building the Neural Network but this time we simply use the original dataset to do it.



Building Neural Network

Click Classify

Click Choose

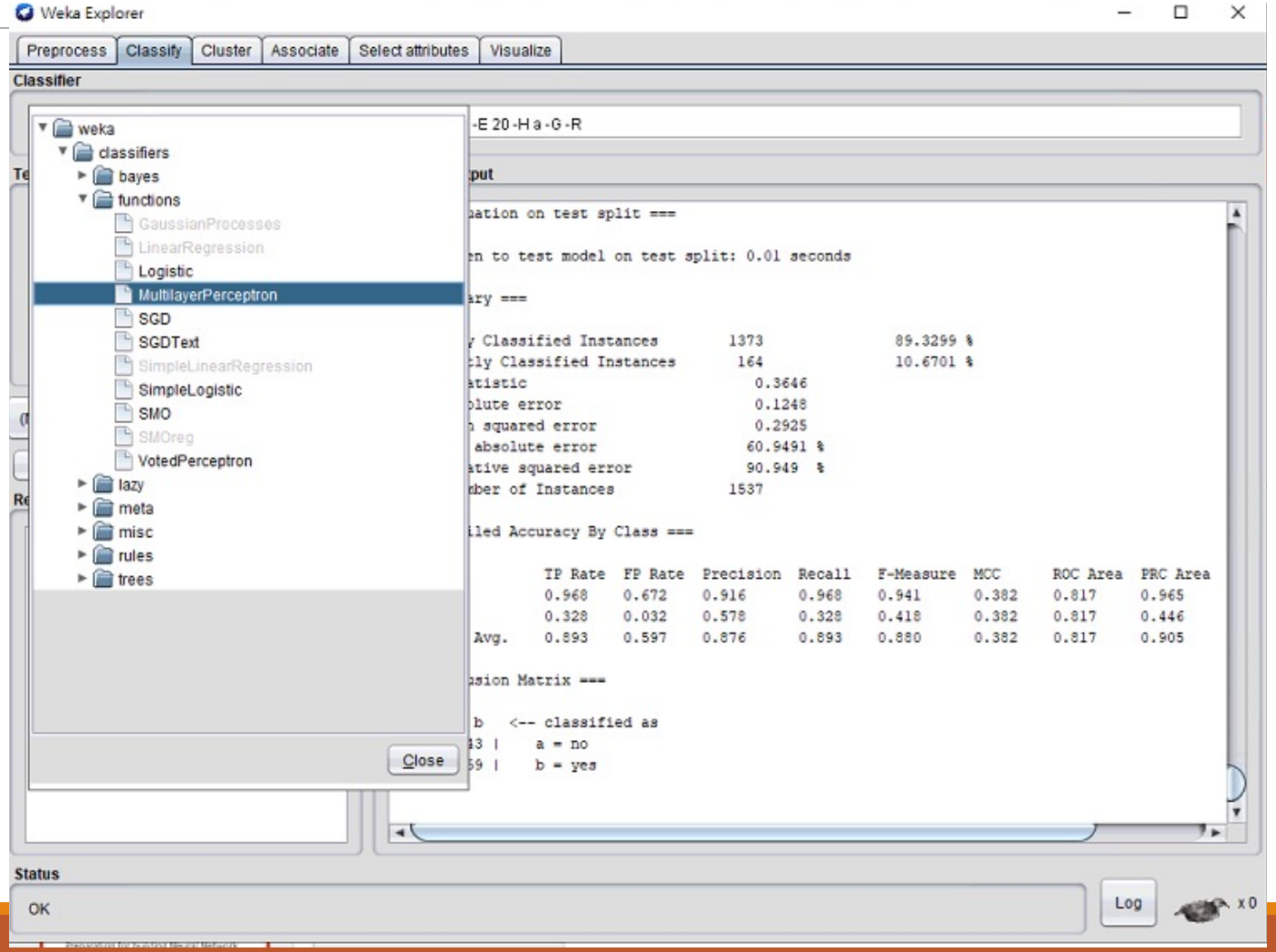


Building Neural Network

Under

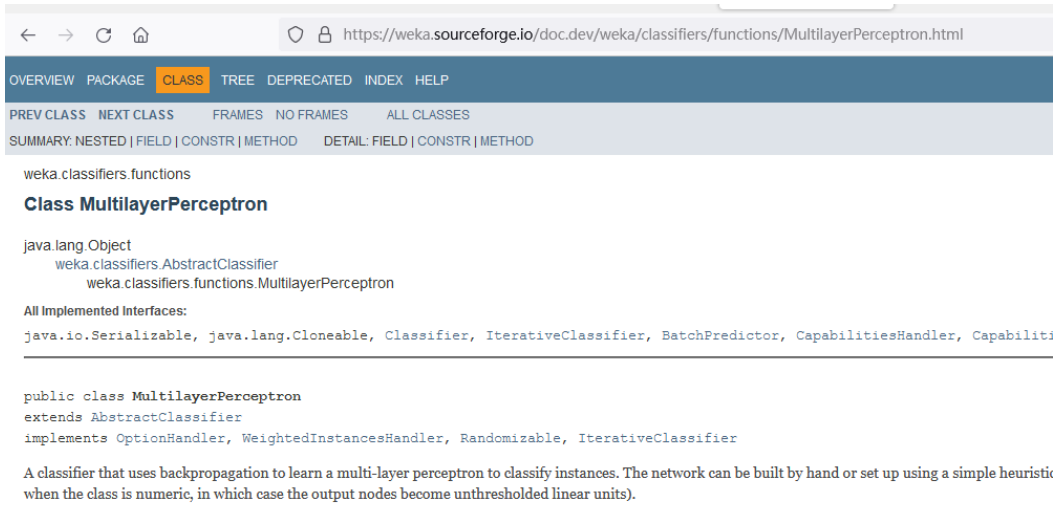
classifiers->functions

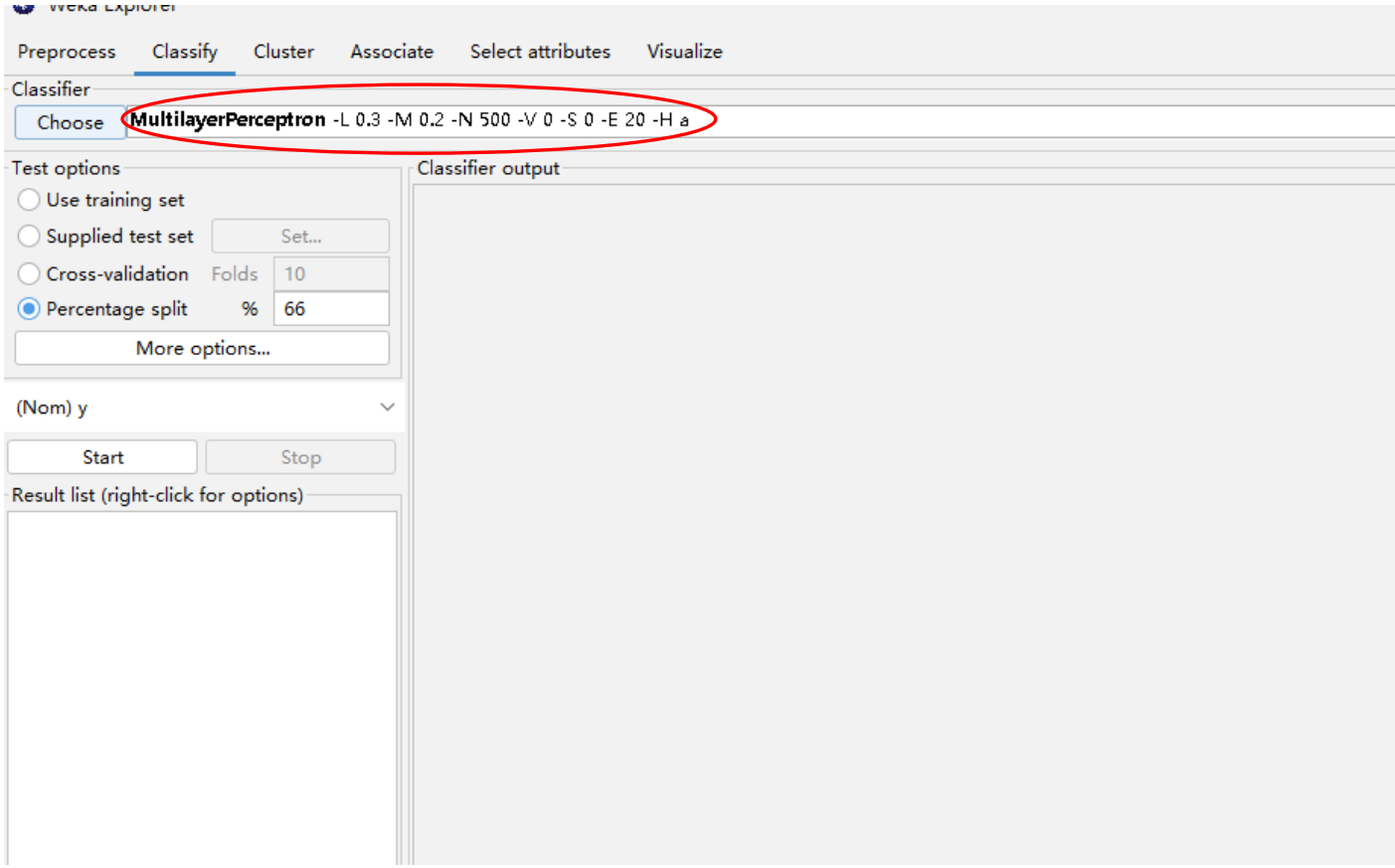
select MultilayerPerceptron



Building Neural Network

Click on the text near Choose to access to the configuration





Valid options are:

```
-L <learning rate>
 Learning rate for the backpropagation algorithm.
 (Value should be between 0 - 1, Default = 0.3).

-M <momentum>
 Momentum rate for the backpropagation algorithm.
 (Value should be between 0 - 1, Default = 0.2).

-N <number of epochs>
 Number of epochs to train through.
 (Default = 500).

-V <percentage size of validation set>
 Percentage size of validation set to use to terminate
 training (if this is non zero it can pre-empt num of epochs.
 (Value should be between 0 - 100, Default = 0).

-S <seed>
 The value used to seed the random number generator
 (Value should be >= 0 and and a long, Default = 0).

-E <threshold for number of consecutive errors>
 The number of consecutive increases of error allowed for validation
 testing before training terminates.
 (Value should be > 0, Default = 20).

-G
 GUI will be opened.
 (Use this to bring up a GUI).

-A
 Autocreation of the network connections will NOT be done.
 (This will be ignored if -G is NOT set)

-B
 A NominalToBinary filter will NOT automatically be used.
 (Set this to not use a NominalToBinary filter).

-H <comma separated numbers for nodes on each layer>
 The hidden layers to be created for the network.
 (Value should be a list of comma separated Natural
 numbers or the letters 'a' = (attribs + classes) / 2,
 'i' = attribs, 'o' = classes, 't' = attribs .+ classes)
 for wildcard values, Default = a).

-C
 Normalizing a numeric class will NOT be done.
 (Set this to not normalize the class if it's numeric).

-I
 Normalizing the attributes will NOT be done.
 (Set this to not normalize the attributes).

-R
 Reseting the network will NOT be allowed.
 (Set this to not allow the network to reset).

-D
 Learning rate decay will occur.
 (Set this to cause the learning rate to decay).
```

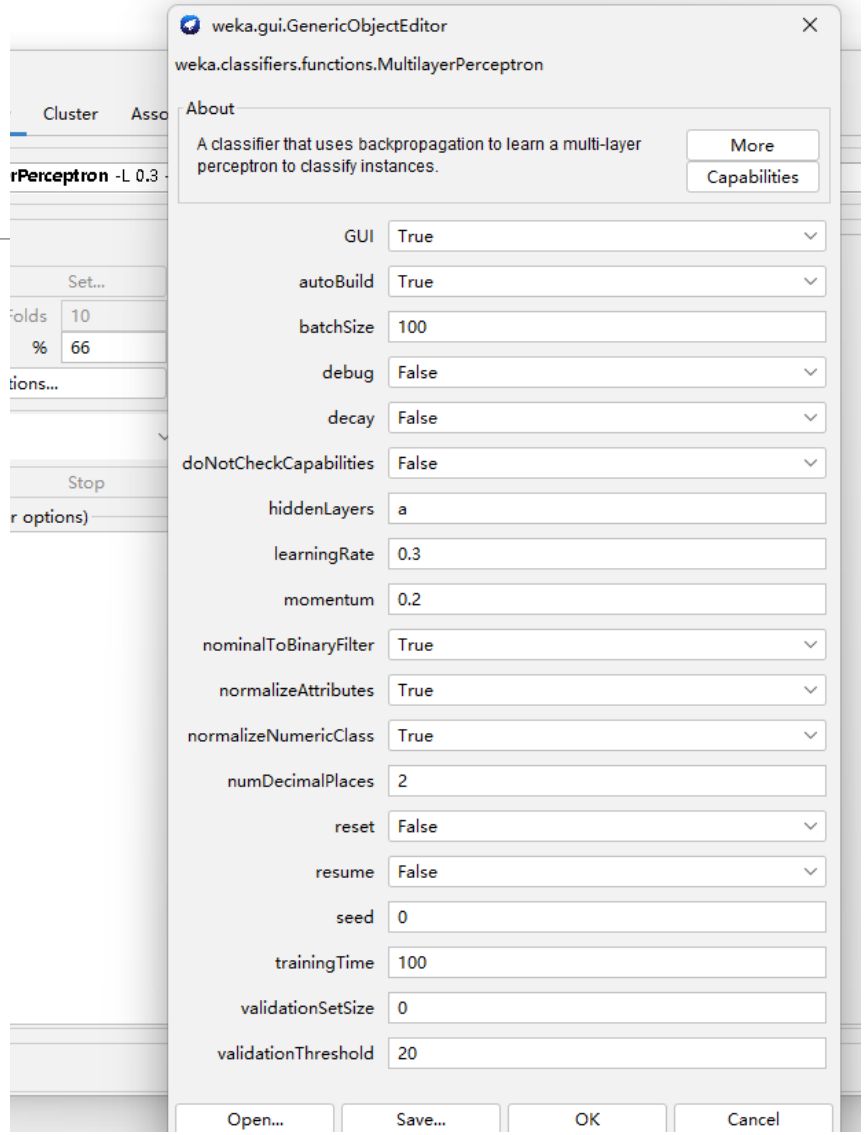
Building Neural Network

Here is the configuration of Multilayer Perceptron.

The default value of **HiddenLayers** is “a” which means Weka will help you to setup the hidden layers. You can also specify how many layer and how many nodes of each hidden layer. For example, type in 10,5,2 means 3 hidden layers with 10, 5, 2 nodes respectively.

trainingTime means how many iterations we want to train through. Let set it from 500 to 100.

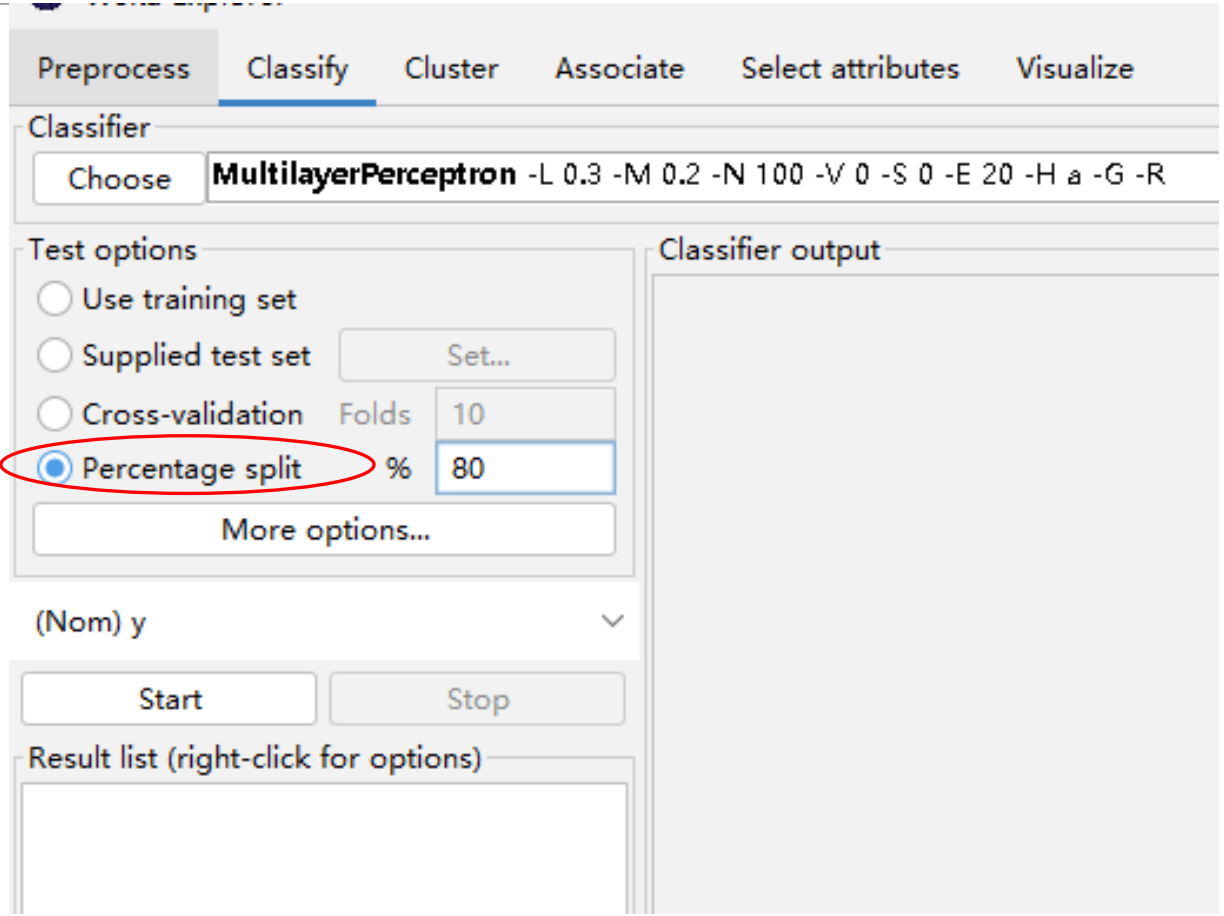
Then, click OK



Building Neural Network

In the Test options here, we simply use percentage split 80% as our testing option.

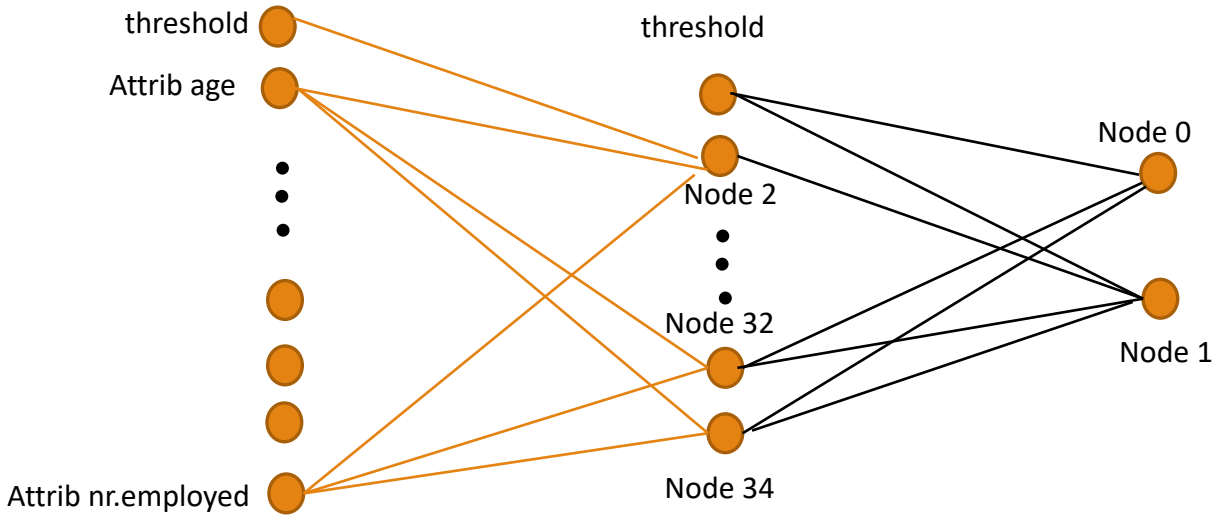
Using 80% of the dataset for training and 20% for evaluation



Building Neural Network

- Click Start to start our neural network training
- Since neural network requires much more computation power compared with decision tree and logistic regression. We need to wait Weka to train our model. The training time depends on the number of parameters (number of layers and number of nodes in each layer), number of iterations and number of data we have.

Interpreting the output



Variable Transformation

Convert Nominal Attributes to Dummy Variables

Some machine learning algorithms prefer to use real-valued inputs and do not support nominal or ordinal attributes.

Nominal attributes can be converted to real values. This is done by creating one new binary attribute for each category. For a given instance that has a category for that value, the binary attribute is set to 1 and the binary attributes for the other categories is set to 0. This process is called creating dummy variables.

1. architecture of the network

2. . Nominal Attributes are converted to Dummy Variables

Interpreting the output

After the training is finished. The result is shown on the right panel.

```
=== Evaluation on test split ===

Time taken to test model on test split: 0.04 seconds

=== Summary ===

Correctly Classified Instances         734               89.0777 %
Incorrectly Classified Instances        90               10.9223 %
Kappa statistic                          0.4053
Mean absolute error                      0.117
Root mean squared error                  0.3124
Relative absolute error                 58.2739 %
Root relative squared error             96.4335 %
Total Number of Instances              824

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.957    0.602    0.922      0.957    0.939      0.412    0.854     0.976     no
               0.398    0.043    0.557      0.398    0.464      0.412    0.854     0.443     yes
Weighted Avg.  0.891    0.536    0.878      0.891    0.883      0.412    0.854     0.913

=== Confusion Matrix ===

   a   b   <-- classified as
 695  31 |   a = no
  59  39 |   b = yes
```

```
=== Evaluation on test split ===

Time taken to test model on test split: 0.04 seconds

=== Summary ===

Correctly Classified Instances         734               89.0777 %
Incorrectly Classified Instances        90               10.9223 %
Kappa statistic                          0.4053
Mean absolute error                      0.117
Root mean squared error                  0.3124
Relative absolute error                 58.2739 %
Root relative squared error             96.4335 %
Total Number of Instances              824

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.957    0.602     0.922      0.957    0.939      0.412    0.854     0.976     no
                 0.398    0.043     0.557      0.398    0.464      0.412    0.854     0.443     yes
Weighted Avg.    0.891    0.536     0.878      0.891    0.883      0.412    0.854     0.913

=== Confusion Matrix ===

   a   b   <-- classified as
 695  31 |   a = no
  59  39 |   b = yes
```

You can omit these evaluation metrics in the red rectangle since they are generally used for the evaluation of regression model instead of classification model.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ – *predicted value of* y

$\bar{y}$ – *mean value of* y

The RMSE can be calculated by taking the square root of above mentioned **Mean Squared Errors (MSE) / L2 Loss**.

Why not use mean squared error for classification problems?

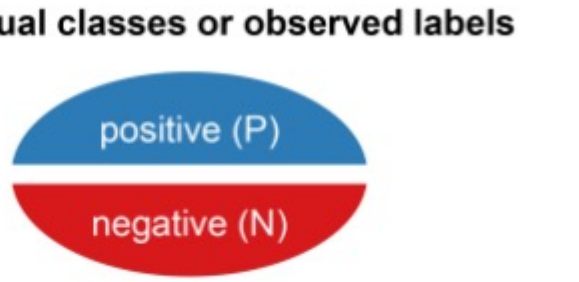
I would like to show it using an example. Assume a 6 class classification problem.

Assume, True probabilities = [1, 0, 0, 0, 0, 0]

Case 1: Predicted probabilities = [0.2, 0.16, 0.16, 0.16, 0.16, 0.16]

Case 2: Predicted probabilities = [0.4, 0.5, 0.1, 0, 0, 0]

The MSE in the Case1 and Case 2 is **0.768** and **0.62** respectively.
Although, Case 1 is correctly predicting class 1 for the instance, the loss in Case 1 is higher than the loss in Case 2.



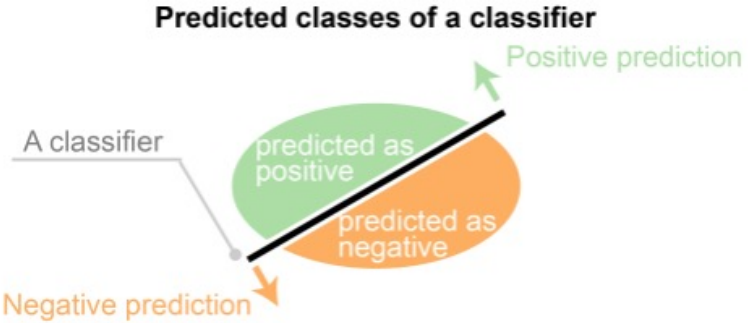
In binary classification, a test dataset has two labels; positive and negative.

The class of interest is usually denoted as “positive” and the other as “negative”.

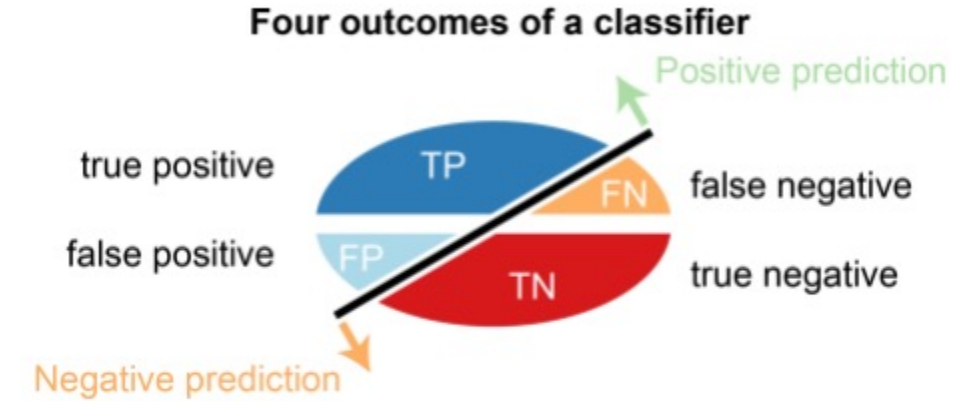
So in this task, yes should be positive or negative?

yes: client subscribed a term deposit

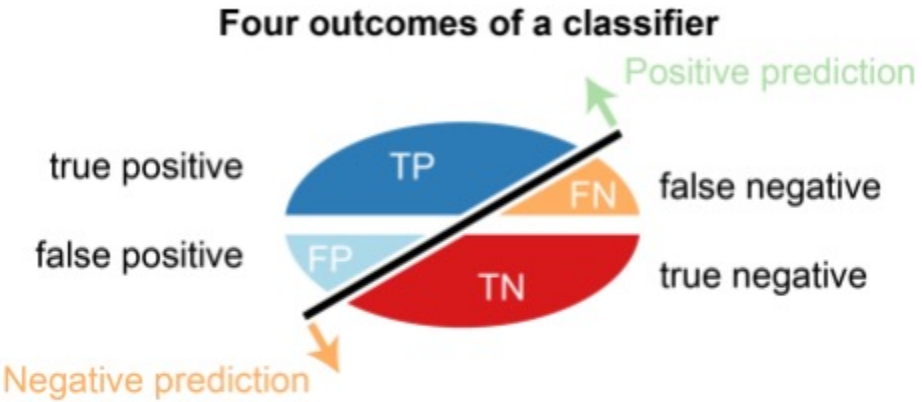
no: client does not subscribe a term deposit



The predicted labels of a classifier match with part of the observed labels.



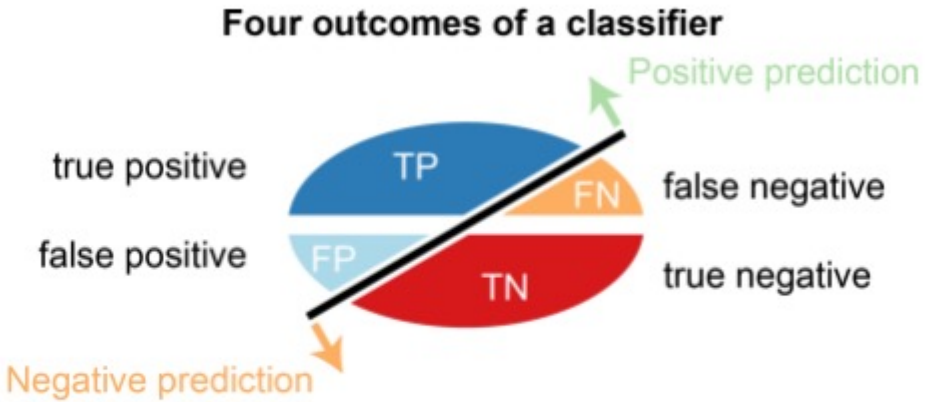
Classification of a test dataset produces four outcomes – true positive, false positive, true negative, and false negative.



Classification of a test dataset produces four outcomes – true positive, false positive, true negative, and false negative.

		Predicted	
		Positive	**Negative**
Observed	**Positive**	TP (# of TPs)	FN (# of FNs)
	Negative	FP (# of FPs)	TN (# of TNs)

Confusion matrix



Classification of a test dataset produces four outcomes – true positive, false positive, true negative, and false negative.

```
=== Evaluation on test split ===

Time taken to test model on test split: 0.04 seconds

=== Summary ===

Correctly Classified Instances         734               89.0777 %
Incorrectly Classified Instances        90               10.9223 %
Kappa statistic                          0.4053
Mean absolute error                      0.117
Root mean squared error                  0.3124
Relative absolute error                 58.2739 %
Root relative squared error             96.4335 %
Total Number of Instances              824

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.957     0.602     0.922     0.957     0.939      0.412    0.854     0.976     no
               0.398     0.043     0.557     0.398     0.464      0.412    0.854     0.443     yes
Weighted Avg.  0.891     0.536     0.878     0.891     0.883      0.412    0.854     0.913

=== Confusion Matrix ===

   a   b   <-- classified as
 695  31 |   a = no
  59  39 |   b = yes
```

If class no is the positive class, this row
TP=695, FN=31,FP=59,TN=39

TP Rate = TP/(TP+FN) = 695/(695+31)=0.957
FP Rate = FP/(FP+TN) = 59/(59+39) = 0.602
Precision = TP/(TP+FP) = 695/(695+59) = 0.922
Recall = TP Rate
F-Measure= 2*Precision*Recall/(Precision+Recall) = 1.765/1.879=0.939

If class yes is the positive class, this row
TP=39, FN=59,FP=31,TN=695

Weighted Avg. TP (695+31)/824*0.957 +(59+39)/824*0.398 = 0.891

Refer to this link: https://classeval.wordpress.com/introduction/basic-evaluation-measures/

Using which metric to judge two classification models?

It depends on the task

Accuracy is usually not a good metric when the dataset is unbalanced (as bank-additional)

In cancer detection system (Check whether a person has cancer), recall is a better evaluation metric. (A higher recall means more cancer patients have been detected)

In email detection system (Detect whether it is a spam email), precision is a better evaluation metric. (We would rather mark spam as normal mail than put normal mail directly into the dustbin)

Under what circumstances F1 OR ROC is better?

https://www.analyticsvidhya.com/blog/2020/10/how-to-choose-evaluation-metrics-for-classification-model/

Save Neural Network Model

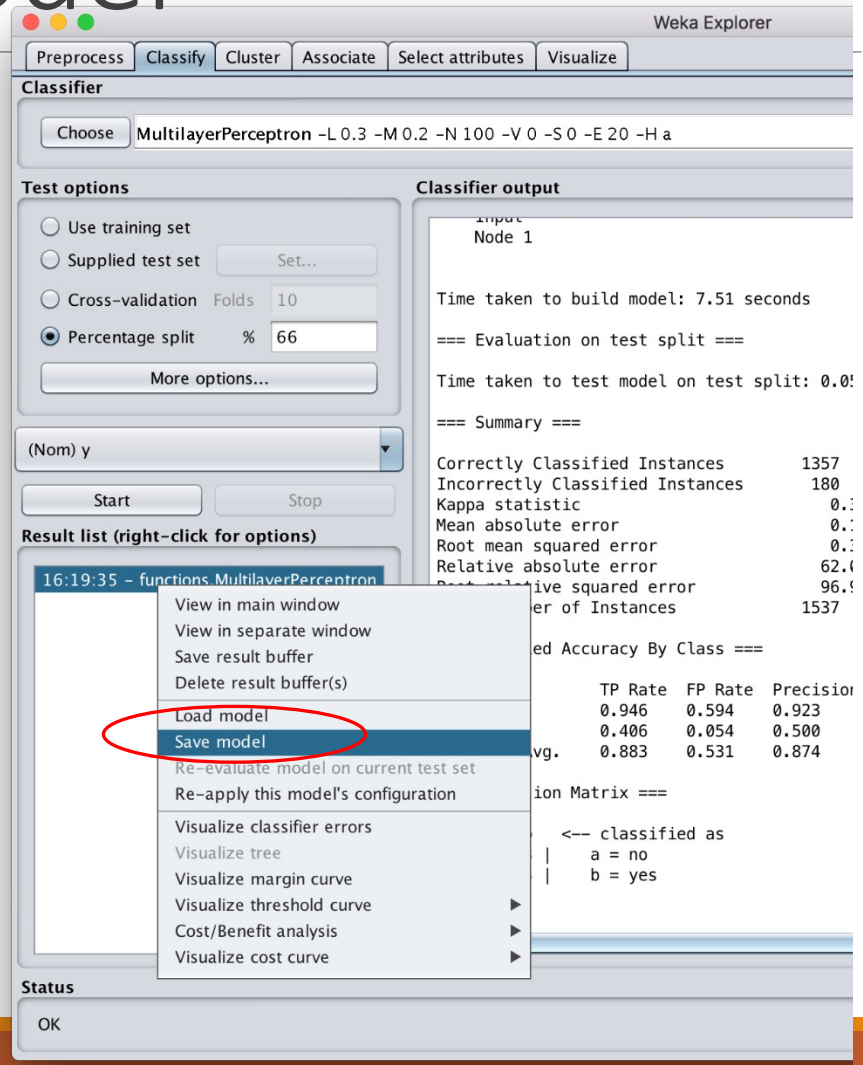
Suppose we want to save the trained multilayer perceptron model.

In the result list, right click the model

Click Save model

Select a location and enter a filename such as mlp, click

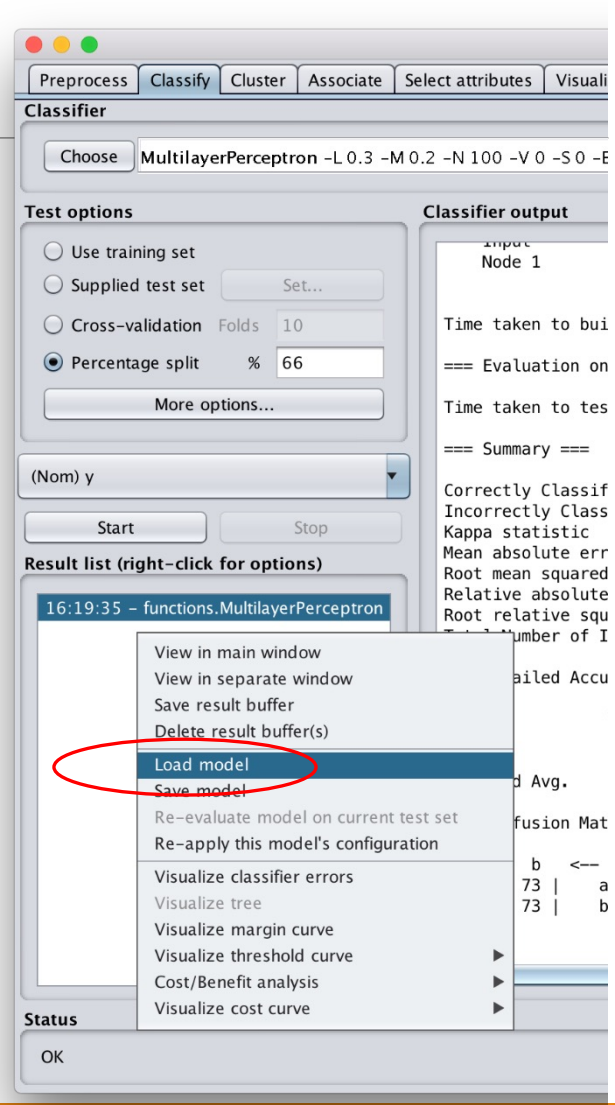
Save Our model is now saved to the file ”mlp.model".



Load Neural Network Model

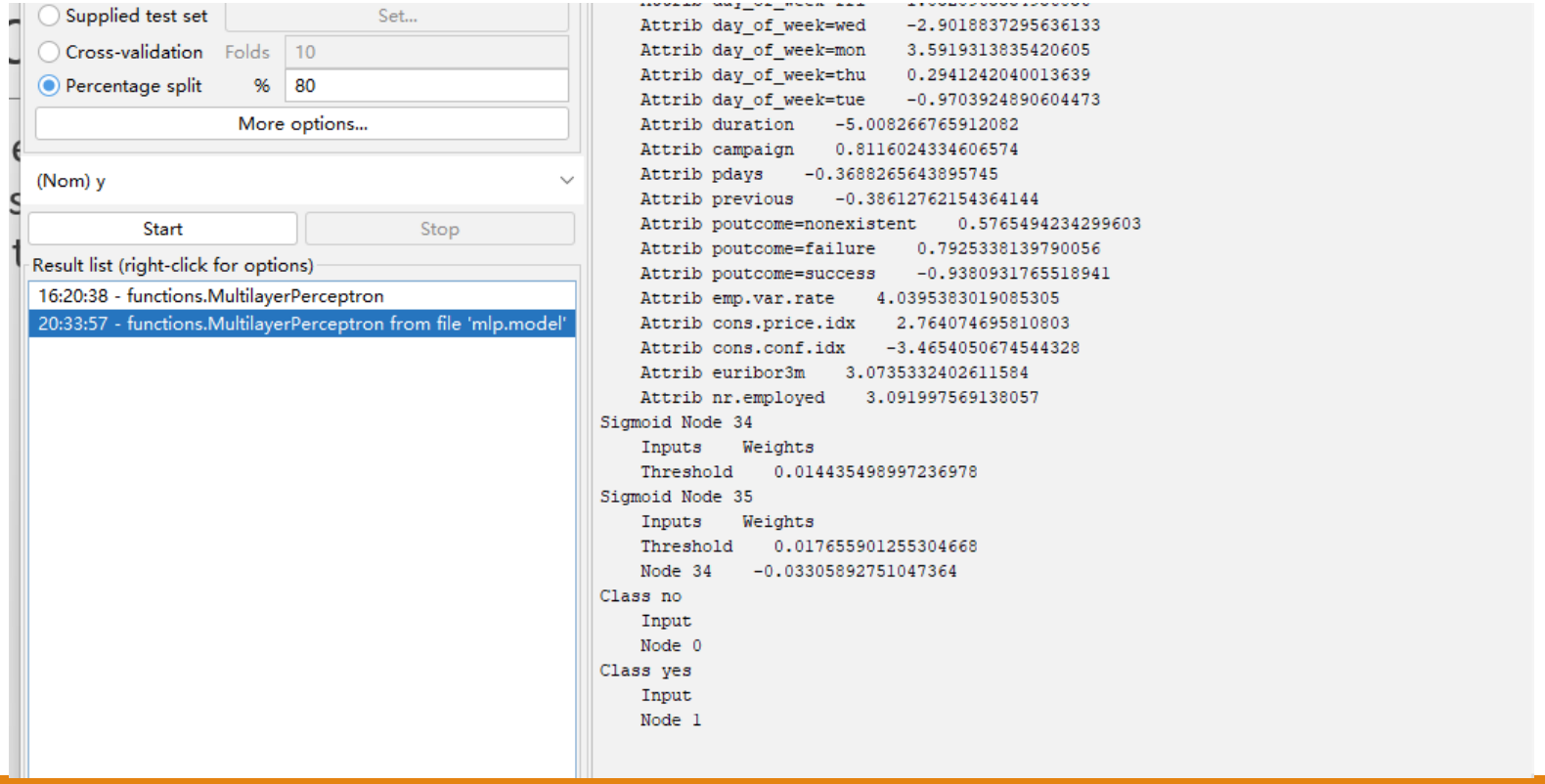
Suppose we want to use our trained model to make prediction.

Right click on the Result list and click Load model, select the model saved in the previous slide ”mlp.model".



Load Neural Network Model

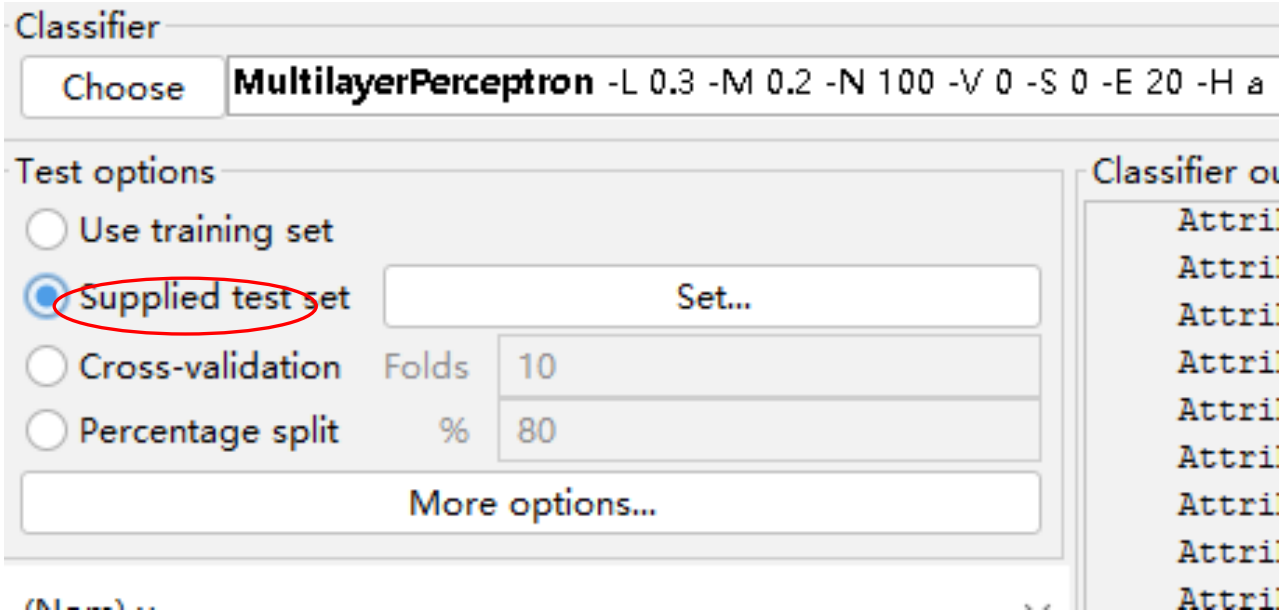
Now, the model is loaded, and we can see some information on the right panel.



Evaluate Model on New Data

We want to evaluate our model on a new dataset.

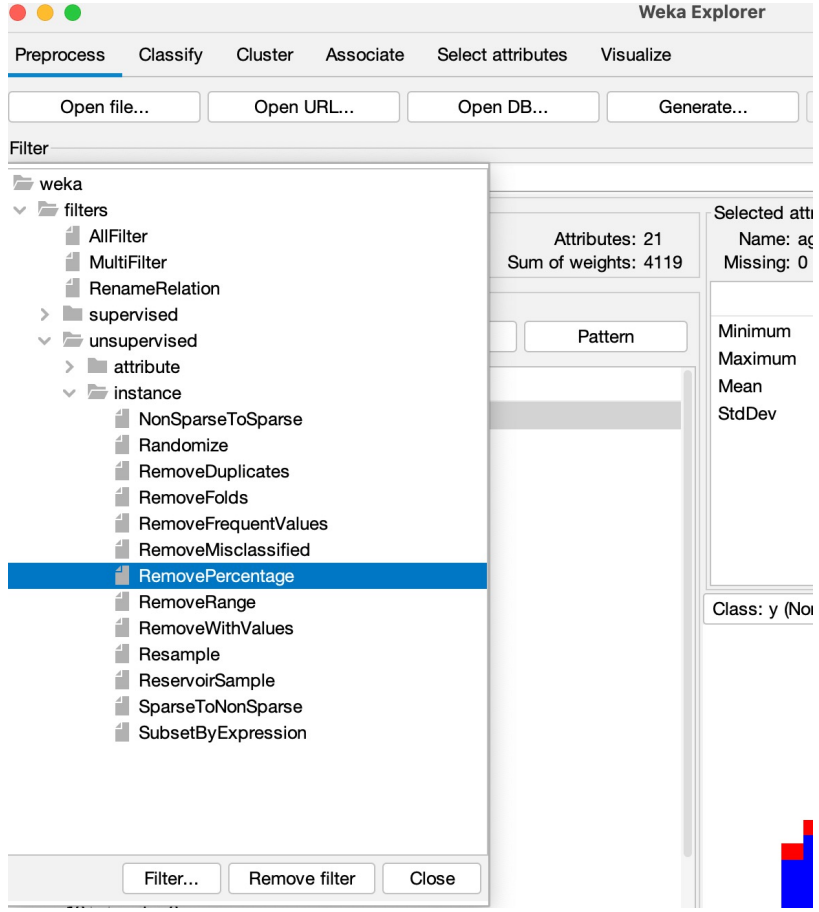
Select the Supplied test set option in the Test options pane.



Evaluate Model on New Data

You can use the RemovePercentage filter to split a dataset. ("bank-additional-test.arff")

- training set:
  - Load the full dataset
  - select the `RemovePercentage` filter in the preprocess panel
  - set the correct percentage for the split
  - apply the filter
  - save the generated data as a new file
- test set:
  - Load the full dataset (or just use undo to revert the changes to the dataset)
  - select the `RemovePercentage` filter if not yet selected
  - set the `invertSelection` property to true
  - apply the filter
  - save the generated data as new file

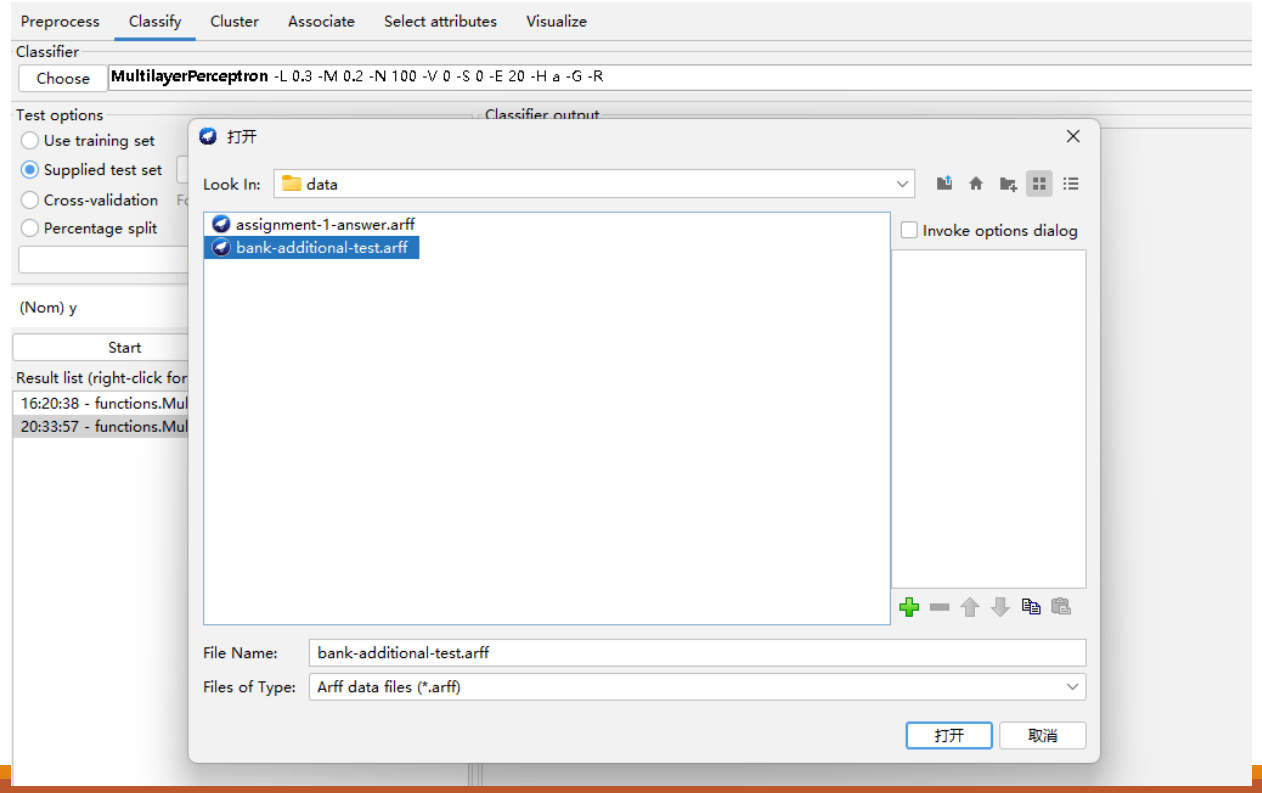


Evaluate Model on New Data

Click Set, click the Open file on the options window and select the new dataset we just created with the name "bank-additional-test.arff".

For the Class, select y

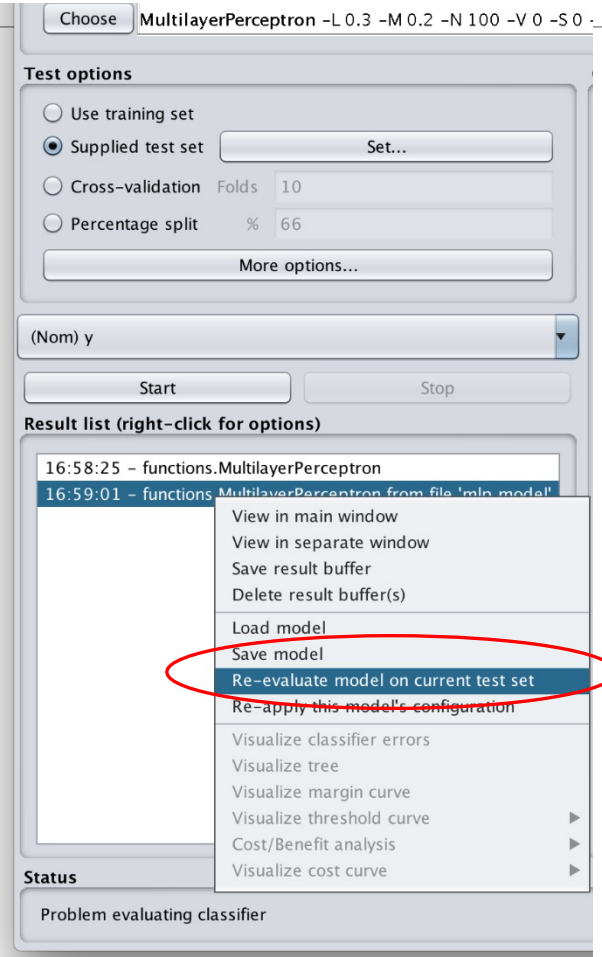
Then, Click Close



Evaluate Model on New Data

Right click on the list item for our loaded model in the Results list.

Choose Re-evaluate model on current test set



Evaluate Model on New Data

After the evaluation is finished. The result is shown on the right panel.

Classifier output

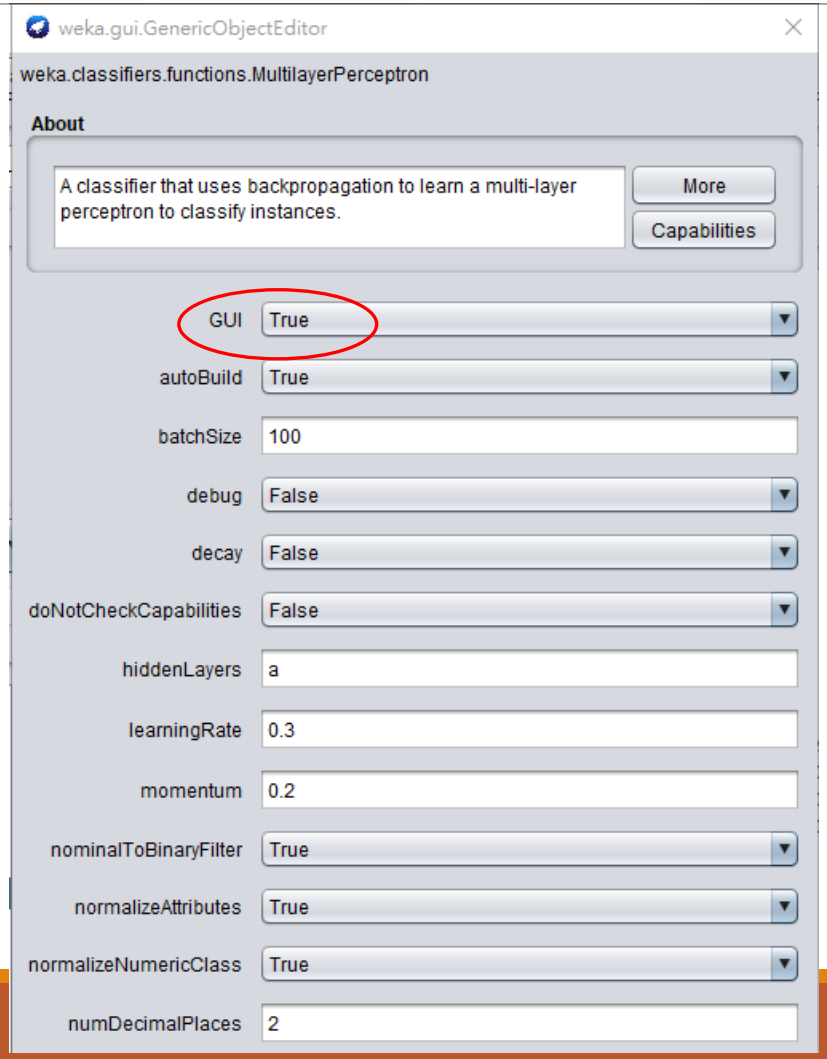
```
    Node 1


=== Re-evaluation on test set ===

User supplied test set
Relation:     bank-additional-new
Instances:     unknown (yet). Reading incrementally
Attributes:   21

=== Summary ===

Correctly Classified Instances        1414               70.7    %
Incorrectly Classified Instances       586               29.3    %
Kappa statistic                          0.414
Mean absolute error                      0.2906
Root mean squared error                  0.5049
Total Number of Instances             2000

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Ar
               0.955    0.541    0.638      0.955    0.765      0.477    0.876     0.863
               0.459    0.045    0.911      0.459    0.610      0.477    0.876     0.868
Weighted Avg.  0.707    0.293    0.775      0.707    0.688      0.477    0.876     0.866

=== Confusion Matrix ===

   a   b   <-- classified as
 955  45 |   a = no
 541 459 |   b = yes
```

Building Neural Network using GUI

Let us investigate more configuration of the neural network.

Change GUI from False to True. This will provide a GUI windows after clicking Start.

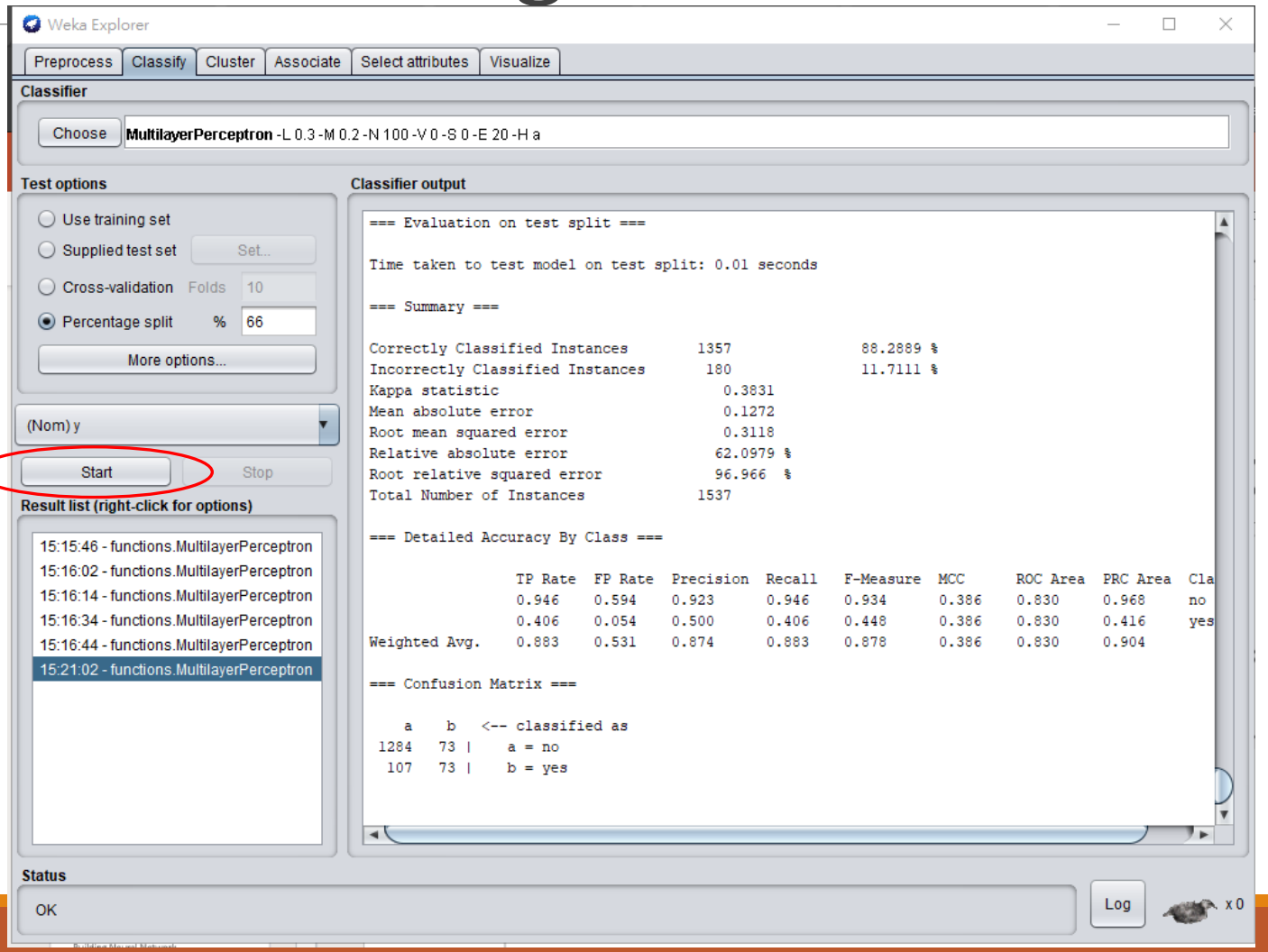
Click OK to close the configuration.



Building Neural Network using GUI

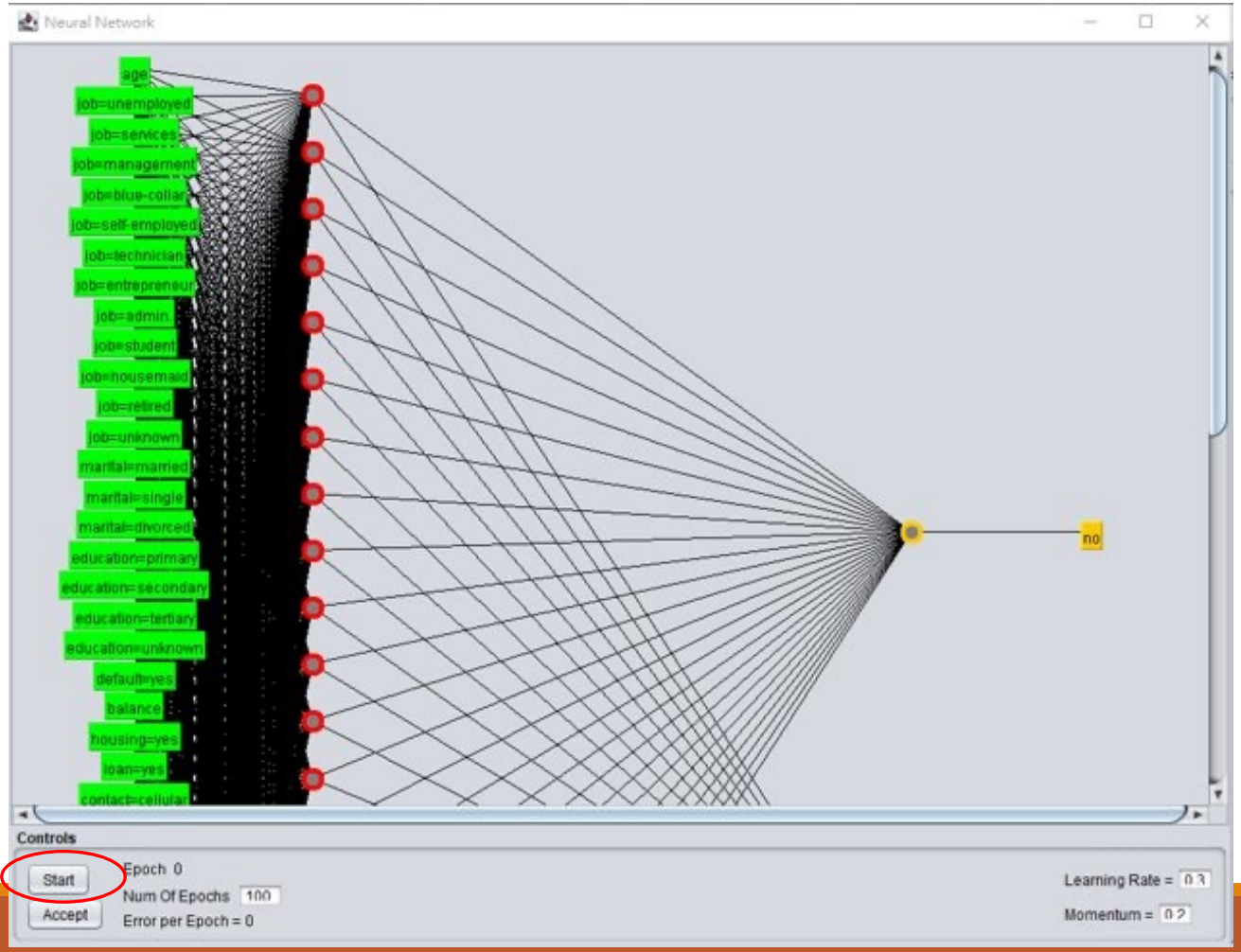
Leave other setting as the same as previous slides.

Click Start



Building Neural Network using GUI

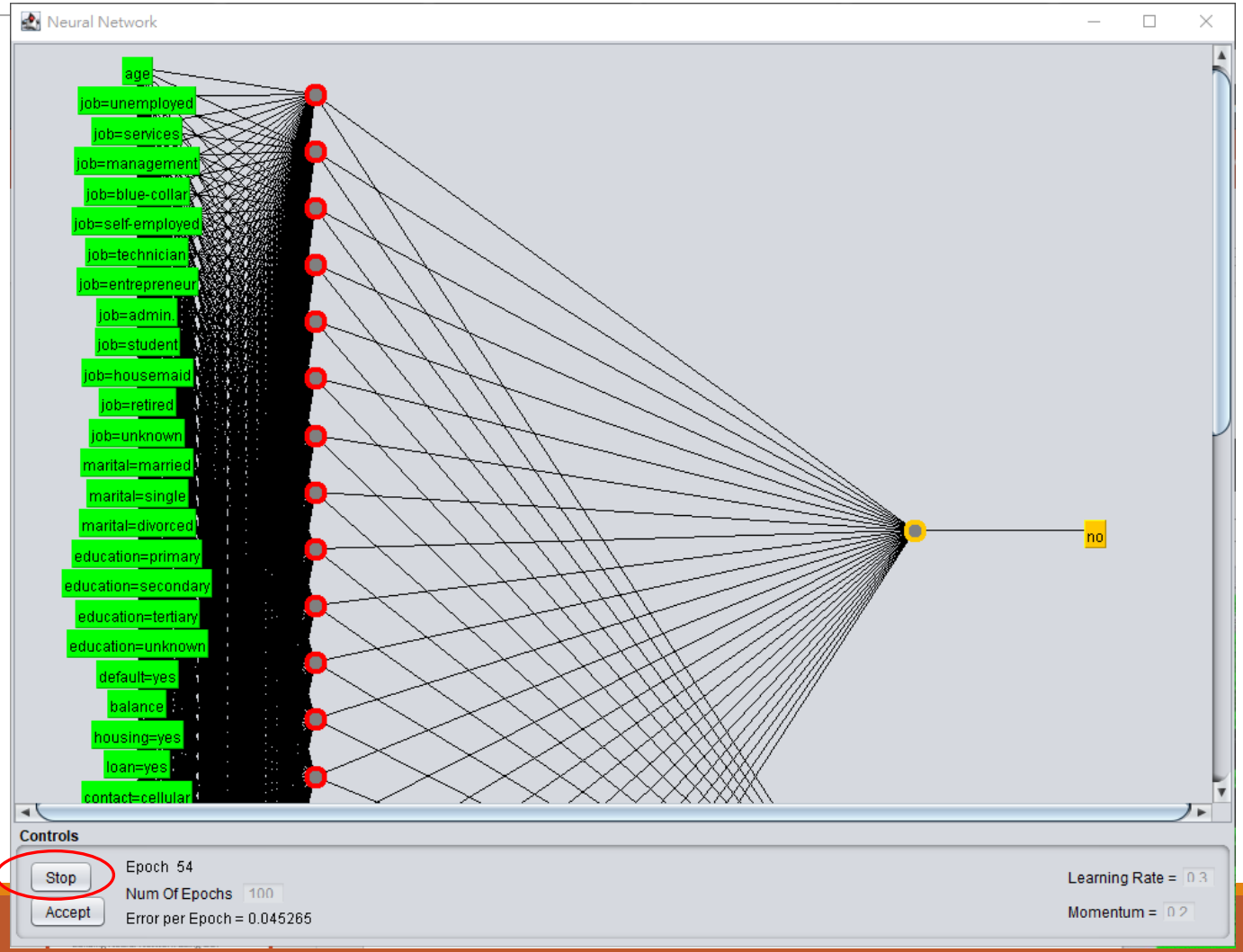
You can click Start to start training.



Building Neural Network using GUI

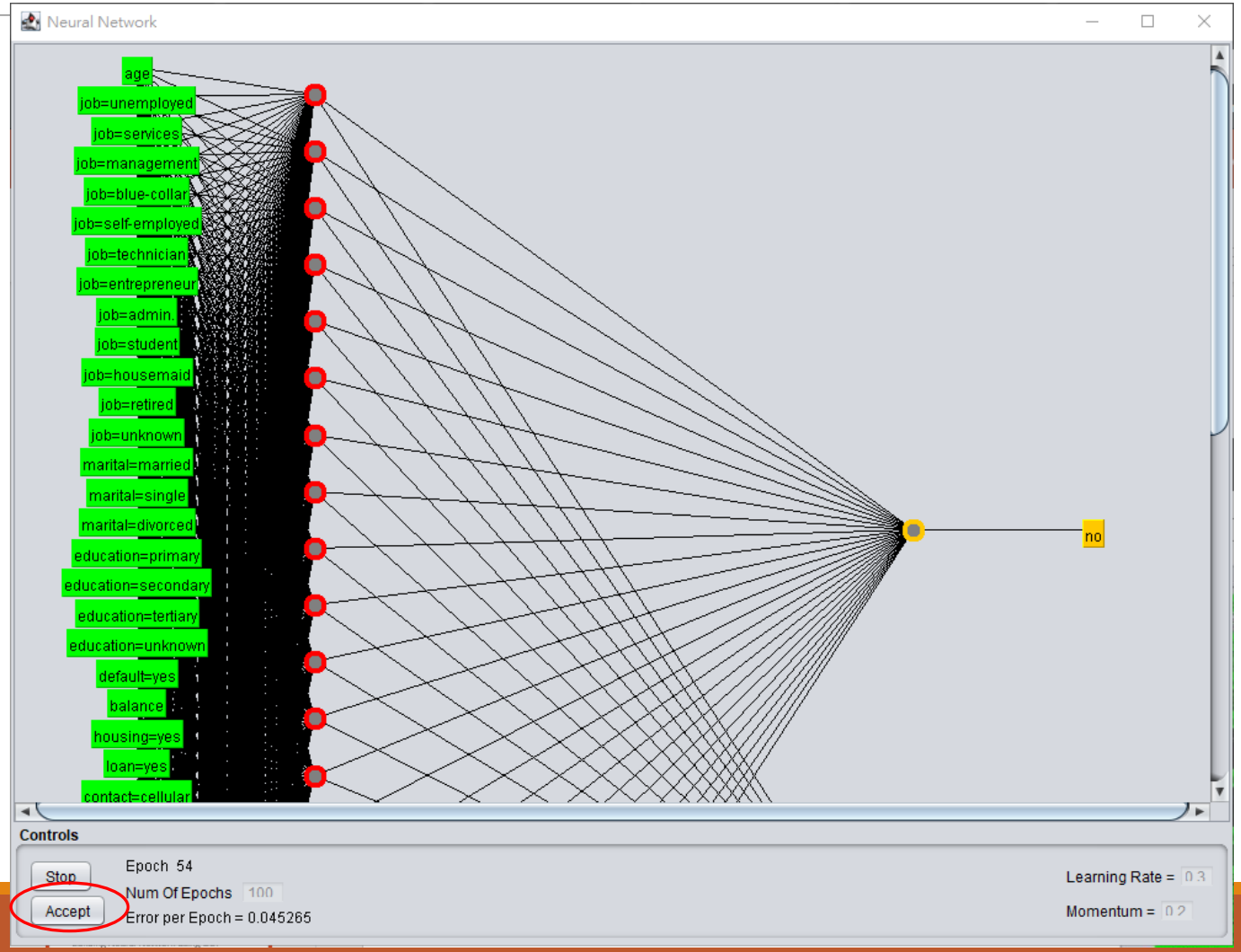
You can click Stop to stop training.

Also, the current epoch and error per epoch will be updated continuously after each epoch.



Building Neural Network using GUI

You can click Accept to finish training although the training has not reached the 100 epochs.



Building Neural Network using GUI

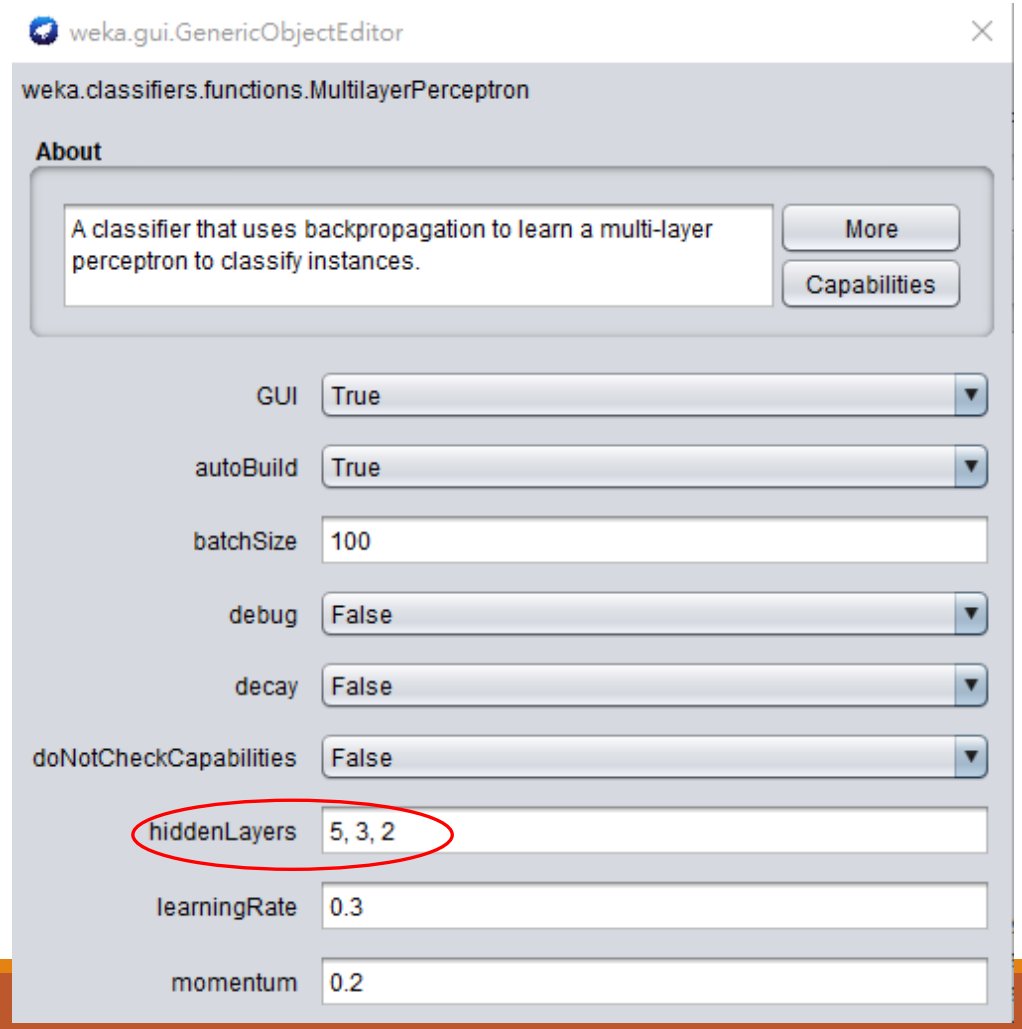
After you click Accept, the result will be shown on the right panel, just like previous slides.

Building Neural Network using GUI

As mentioned in previous slides, you can specify the number of hidden layers and the nodes of each layer. If you turn on the GUI mode, you can easily to verify it.

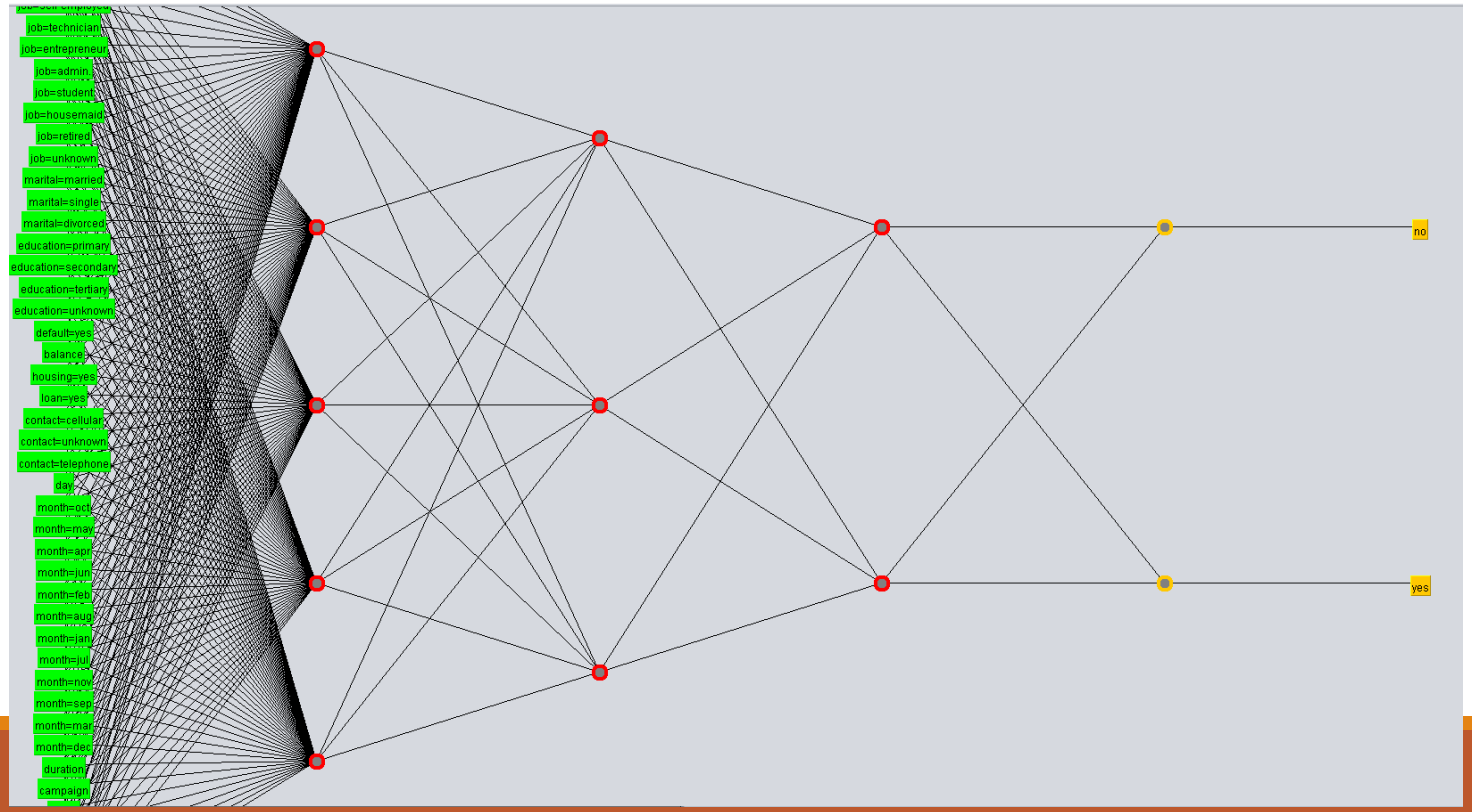
Let's change the hiddenLayers to 5,3,2

Click OK



Building Neural Network using GUI

After clicking Start, you can see now your network has 3 hidden layers where 5 nodes in layer 1, 3 nodes in layer 2 and 2 nodes in layer 3.



Remarks on Neural Network

The performance of the neural network can be easily affected by the setup of the **hyperparameters**. The hyperparameters include the number of hidden layers, number of nodes, learning rate, momentum, batch size, etc. To achieve a better performing neural network always requires tons of hyperparameters tuning. You can play with different hyperparameters setting and investigate which combination can achieve a better result.

20-min practice, QA

Clustering Algorithms with Weka

A clustering algorithm finds groups of similar instances in the entire dataset. WEKA supports several clustering algorithms such as EM, FilteredClusterer, HierarchicalClusterer, SimpleKMeans and so on. You should understand these algorithms completely to fully exploit the WEKA capabilities.

As in the case of classification, WEKA allows you to visualize the detected clusters graphically.

Clustering Algorithms with Weka

After loading data into Explorer, click Cluster.

If the dataset has a label, you need transform the label to nominal attribute (if the label originally is a numeric attribute)

Click Choose

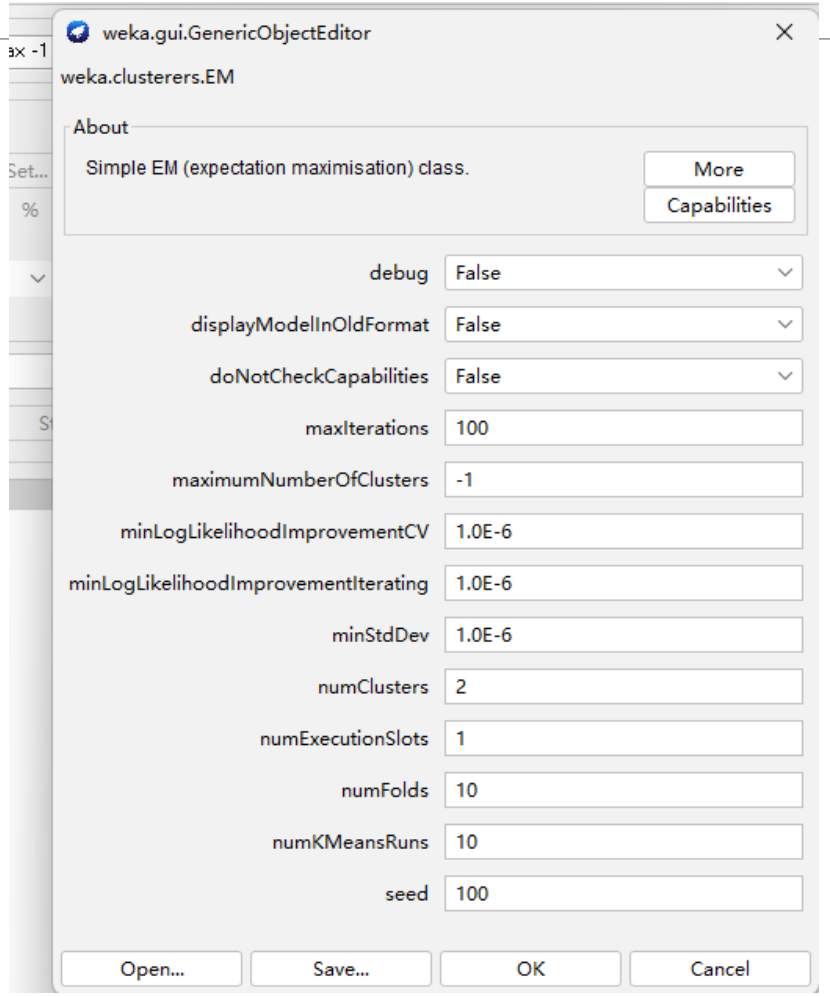
Under clusterers choose EM

Clustering Algorithms with Weka

Set the numClusters to 2.

In the **Cluster mode** sub window, select the **Classes to clusters evaluation** option.

Click on the **Start** button to process the data. After a while, the results will be presented on the screen.



Clustering Algorithms with Weka

From the output screen, you can observe that –

There are 2 clustered instances detected in the database.

```
Time taken to build model (full training data) : 0.13 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       1717 ( 42%)
1       2402 ( 58%)


Log likelihood: -38.72291


Class attribute: y
Classes to Clusters:

    0    1  <-- assigned to cluster
 1385 2283 | no
  332  119 | yes

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances :       1504.0   36.5137 %
```

From the output screen, you can observe that –

For the nomimal attributes, each type of a specific attribiute is assigned a value.

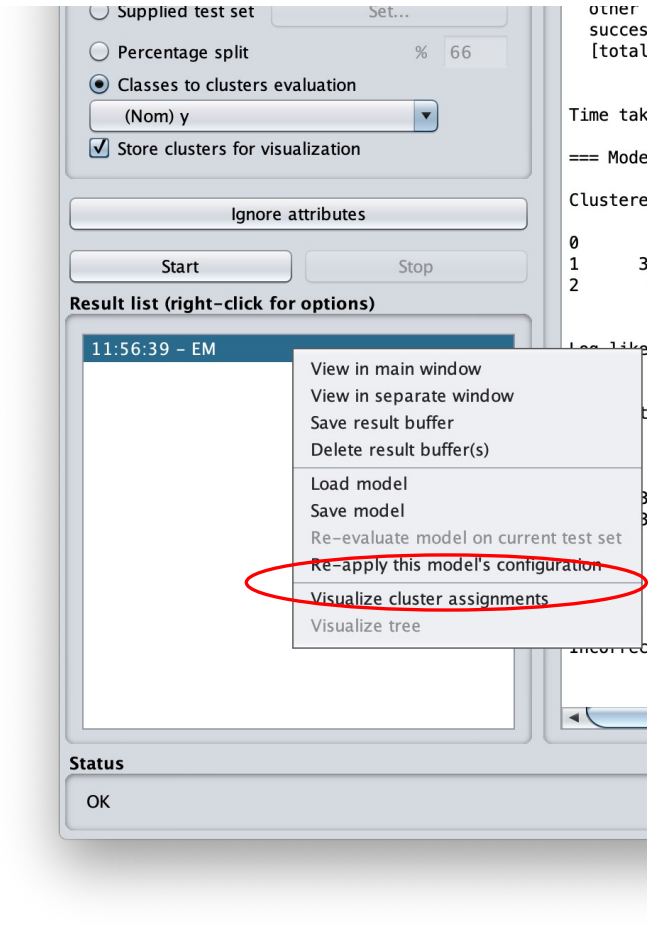
For the numerical attributeds some statistics like the mean and standard deviation are given.

Attribute	Cluster 0 (0.45)	1 (0.55)
age		
mean	39.9539	40.2426
std. dev.	11.3329	9.4056
job		
blue-collar	296.8006	589.1994
services	152.5899	242.4101
admin.	533.4917	480.5083
entrepreneur	63.6811	86.3189
self-employed	73.1585	87.8415
technician	289.4264	403.5736
management	180.7106	145.2894
student	60.3143	23.6857
retired	97.305	70.695
housemaid	36.7198	75.2802
unemployed	56.0478	56.9522
unknown	11.6538	29.3462
[total]	1851.8996	2291.1004
marital		
married	1025.1569	1485.8431
single	621.3357	533.6643
divorced	191.2261	256.7739
unknown	6.1808	6.8192
[total]	1843.8996	2283.1004
education		
basic.9y	210.7787	365.2213
high.school	393.0232	529.9768
previous		
mean	0.4258	0.0003
std. dev.	0.746	0.0164
poutcome		
nonexistent	1245.5124	2279.4876
failure	454.3872	1.6128
success	143	1
[total]	1842.8996	2282.1004
emp.var.rate		
mean	-1.309	1.2103
std. dev.	1.3353	0.3747
cons.price.idx		
mean	93.1373	93.9369
std. dev.	0.4983	0.3471
cons.conf.idx		
mean	-41.8112	-39.4398
std. dev.	5.7312	3.019
euribor3m		
mean	2.0759	4.869
std. dev.	1.5275	0.2491
nr.employed		
mean	5107.9808	5213.709
std. dev.	74.3682	18.6448

Visualize Clusters with Weka

To visualize the clusters, right click on the **EM** result in the **Result list**. You will see the following options –

Select **Visualize cluster assignments**.



Explain Assignment2

20-min practice, QA