Diagnosis and Classification of Type II Diabetes based on Multilayer Neural Network

Ekhlas S. Nasser*, Faten Abd Ali Dawood

Computer Science Department, College of Science, University of Baghdad, Baghdad, Iraq

Received: 23/2/2021          Accepted: 17/4/2021

Abstract

Diabetes is considered by the World Health Organization (WHO) as a main health problem globally. In recent years, the incidence of Type II diabetes mellitus was increased significantly due to metabolic disorders caused by malfunction in insulin secretion. It might result in various diseases, such as kidney failure, stroke, heart attacks, nerve damage, and damage in eye retina. Therefore, early diagnosis and classification of Type II diabetes is significant to help physician assessments.

The proposed model is based on Multilayer Neural Network using a dataset of Iraqi diabetes patients obtained from the Specialized Center for Endocrine Glands and Diabetes Diseases. The investigation includes 282 samples, of which 240 are diabetic and 42 are non-diabetic patients. The model consists of three main phases. In the first phase, two steps are applied as a pre-processing for the dataset, which include statistical analysis and missing values handling. In the second phase, feature extraction is used for diabetes Type II using three main features, reflecting measurements of three blood parameters (C. peptide, fasting Blood Sugar, and Haemoglobin A1C). Finally, classification and performance evaluation are implemented using Feed Forward Neural Network algorithm. The experimental results of the performance of the proposed model showed 98.6% accuracy for diabetes classification.

Keywords: Diabetes, Multilayer Neural Network, Accuracy, Classification, Specificity.
The proposed model is a scientific field handling the approaches by which machines are learning from experience. A lot of scientists are assuming that the term “machine learning” is similar to the term “artificial intelligence”, considering that the learning possibility is the major property of an entity termed intelligent in the word’s broadest sense. In ML, the major aim is to construct a computer system with the ability for adapting and learning from experience [7]. In addition, Multilayer Neural Networks (MLNNs) have been efficiently utilized in the medical diagnosis because of their excellent accuracy in recognition, approximation, and classification of problems [8].

Classification is the problem of specifying items in a group into different classes. The classification goal is anticipating the target class correctly for each sample of the data. In ML, classification is considered as an instance of supervised learning, which means the learning process in which a training set of correctly identified observations is available. It is represented by two steps, namely the learning step and classification step. In the learning step (training phase), the construction of classification model is performed, in which different algorithms are used to build a classifier by learning the available training set of the model. The model has to be trained to be able to predict the correct results. In the classification step, a proposed model is used to classify the testing data and thus estimate the accuracy of the classification rules based on predefined class labels [7].

The main aim of the proposed model is to diagnose and classify Type II diabetes in a way that can help the physicians to predict if a subject has diabetes or not, using Feed Forward Neural Network (FFNN) algorithm. To accomplish this aim, a sample of Iraqi patients was used with that algorithm. The statistical measurements are used to analyze the dataset for handling the missing values. Then, Feed Forward Neural Network (FFNN) algorithm is implemented for classification (Training, Validation, and Testing) and performance evaluation.

Three features were used in the proposed model. The first one is C-peptide test which is a measurement of pancreatic beta cell function [9], the second is fasting blood sugar (FBS) which was measured using a glucometer [10], and the third is haemoglobin A1c (HbA1c) that occurs due to the non-enzymatic binding of hexose molecule to N-terminal amino acid in the haemoglobin molecule [11].
2. Related Work

In the literature investigated, several approaches based on machine learning have been proposed for early diagnosis and classification of Type II diabetes mellitus. Most of these approaches have been applied using the public Pima Indians diabetes dataset. Therefore, to our knowledge, our proposed model is the first study which is implemented on an Iraqi diabetes patient’s dataset.

Acar et al. [12] proposed a technique to forecast DM with the use of many ANN types, among which the Multi-Layer Perceptron (MLP) is considered as the most basic and common approach. The dataset collected from the Pima Indian population near Phoenix, Arizona, was used, consisting of 768 persons partitioned randomly into two independent sets for the purposes of training and testing; 268 were diagnosed with diabetes and the remainder were non-diabetic. The results of the accuracy from Pima Indian dataset was 70.15% with 150 epochs and 8-20-1 structure of MLP.

Saumendra et al. [13] proposed a method by using MLNN for the classification of diabetic pregnant women. The suggested approach was utilized on Pima diabetes dataset obtained from the University of California. The data was split into 30% (218 cases) for testing data and 70% (550 cases) for training data. The two datasets were analysed based on eight features and divided into diabetes (1) and non-diabetes (0) classes. The authors reported that a classification accuracy of 77.5% was obtained.

Bani-Salameh et al. [14] proposed a method using MLNN to predict hypertension and diabetes. There were 768 cases in the dataset, 268 were non-diabetic, whereas 500 were diabetic. The dataset of blood pressure disorder included 768 cases, among which 268 were diagnosed as non-hypertensive, whereas 500 were diagnosed with hypertension. The results showed an accuracy value of 77.63%.

Bani-Hani et al. [15] proposed an approach by using optimized recursive general regression NN oracle to predict and diagnose diabetes with the use of MLNN. The suggested model used Pima dataset, based on a research initially carried out by the National Institute of Diabetes and Digestive and Kidney Diseases. The results demonstrated an accuracy value of 76.88% as the best outcome achieved by the system.

3. Materials and methods

The overall framework of the proposed model of diabetes classification in Iraqi patients is based on MLNN. This model includes three major phases, as shown in Figure (1). The first phase is applied for pre-processing, which consists of two main steps; statistical analysis and missing values handling. In the second phase, features extraction is used for diabetes Type II using three main blood features (C. Peptide, fasting Blood Sugar, and Haemoglobin A1C). Finally, classification and performance evaluation are implemented using Feed Forward Neural Network algorithm that includes two steps; Training and Testing.
3.1 Iraqi Diabetes Patients Dataset (IDPD)
The Iraqi diabetes patient’s dataset (IDPD) was obtained from the Specialized Center for Endocrine Glands and Diabetes Diseases in Baghdad, which consisted of 282 samples. The samples included 240 diabetic and 42 non-diabetic subjects. Among several features, three measurements of blood parameters were considered to be linked to Diabetes Type II (C peptide, Fasting Blood Sugar (FBS), and Haemoglobin A1C (HbA1C)). In the “class” variables of this dataset, a value of (0) indicates negative (non-diabetic) test, whereas a value of (1) indicates positive (diabetic) test. The dataset had four variables, divided into the three input variables and one target variable. The features used to classify the dataset, as well as the obtained classes, are provided in Table (1).
3.2 Pre-processing
Pre-processing is the first phase of the suggested model for diabetes Type II classification. It consists of two main steps, statistical analysis and missing values handling, as explained in the following sub-sections.

3.2.1 Statistical Analysis
Five statistical analyses were used in the proposed model to identify the magnitude of variance, as well as standard deviation, in the data. These included minimum value, maximum value, mean, standard deviation, and variance, as explained below and listed in Table (2).

Equation (1) below was used to compute the mean value of each tested parameter [16].
\[
\text{Mean } (x') = \frac{\sum x_i}{n} 
\]

Equation (2) below was used to calculate the standard deviation for the mean values of the tested parameters.
\[
\text{Standard Deviation } = \sqrt{\frac{\sum_{i=1}^{n} (x_i-x')^2}{n-1}}. 
\]

Equation (3) below was used to calculate the variance of the mean values [14].
\[
\text{Variance } \sigma^2 = \frac{\sum_{i=1}^{n} (x_i-x')^2}{n} 
\]

<table>
<thead>
<tr>
<th>Feature variable</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Standard deviation</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- C.peptide</td>
<td>0.01</td>
<td>15.5</td>
<td>3.695</td>
<td>3.091</td>
<td>9.484</td>
</tr>
<tr>
<td>2- Fast Blood Sugar</td>
<td>3.4</td>
<td>29</td>
<td>12.457</td>
<td>5.264</td>
<td>27.498</td>
</tr>
<tr>
<td>3-Haemoglobin A1c</td>
<td>4.9</td>
<td>121.5</td>
<td>9.878</td>
<td>10.121</td>
<td>102.434</td>
</tr>
</tbody>
</table>

As noticed in Table (2), the variance and standard deviation were calculated to know the range of data dispersion for the three features (C.peptide, FBS, and HbA1C). The standard deviation value of the mean values of HbA1C was the highest (10.121 mmol/mol), while the variance for this feature was equal to (102.434 mmol/mol).

3.2.2 Missing Values Handling
The missing values concept must be grasped for effectively managing the data. Missing values occur frequently and might have important impacts on the conclusions that might be drawn from the data. In the absence of proper handling of missing values, imprecise inference of the data might be reached. Also, inadequate handling typically leads to different results. Moreover, in the case of a large number of missing values, these cases should be dropped and replaced, rather than being imputed.
At the same time, in unilateral analysis, the amount of bias in the data might be decreased via imputation, in the case when the values are missing at random. Missing values occur when there is no data value stored for the variable.

In the proposed model, the missing values in the IDPD were manipulated via utilizing the mean method to compute all missing values. Each missing value was replaced by the mean value for the same feature in the data. Table (3) shows an example of these missing values in IDPD dataset, which have zero value for each.

### Table 3-An example of Iraqi Patients Diabetes Dataset with missing values

<table>
<thead>
<tr>
<th>No. of Sequence</th>
<th>C. peptide (ng/mL)</th>
<th>Fasting Blood Sugar (mmol/L)</th>
<th>Hemoglobin A1c (mmol/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.2</td>
<td>11.8</td>
<td>12.9</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.05</td>
<td>8</td>
<td>10.2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>12.5</td>
</tr>
<tr>
<td>5</td>
<td>3.6</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4.7</td>
<td>29</td>
<td>14.8</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Number of missing values with percentage:
- 0 (0 %) for C. peptide
- 19 (6.7%) for Fasting Blood Sugar
- 146 (51.7%) for Hemoglobin A1c

The number of sequence in Table (3) represents only seven randomly selected samples, with missing values, from the total 282 samples of the dataset. In addition, the last row of the table shows the number of missing values and percentage in relation to the total number of samples.

### 3.3 Feature Extraction

The feature extraction phase is the main step in the implementation of the proposed model for diabetes Type II diagnosis and classification. Three main features, as mentioned in Table (1), were obtained from IDPD and used as an input dataset (i.e., feature vector) in the proposed model of MLNN.

### 3.4 Classification using MLNN

MLNN is a type of ANN that has an output layer, input layer, and at least one hidden layer between them. Each one of the nodes in one layer is fully-connected to each node in the next layer. Each node, except for nodes in the input layer, is a neuron (or a processing element) with non-linear activation function connected with a scalar weight that is modulated during training. MLNN is a supervised learning algorithm that learns identifying data patterns for classification or regression. The goal of the MLNN learning process is finding a set of ideal weights over a training dataset for making the outputs as close as possible to the target values [17]. Figure (2) shows the suggested model’s structure with the three layers. The input layer involves the input values for the network and the hidden layer involves the weights and bias for each input in the network. After that, weights are collected with the activation function. Then, the output layer is reached, which is the final result in which the network output is obtained.
Figure 2—Structure of the proposed model using MLNN.

- Input values: The perceptron takes real values as its inputs. Weights and Bias: The weights that represent the relative importance of each of the weights to the classification decision. Weighted Sum: The input values are multiplied by the weights and summed up, to create one aggregate value which is fed into the activation function. Activation Function: The activation function generates classification decision.

The proposed model of MLNN comprises five steps, as follows:

**Step 1:** Initialization, which is conducted for neural network inputs and outputs.

**Step 2:** Finding initial weights with random values where “Rnd()” is a function that returns random values between (0, 1).

**Step 3:** Finding only the maximum that is included in the class number because it is required in other steps.

**Step 4:** NN training to find the weights which are minimizing the prediction error. Typically, the training started with a set of randomly-generated weights. After that, back propagation was utilized for updating the weights for adequately mapping inputs to outputs. Through every epoch (Data type was Integer Domain [1, ∞), whereas the typical value was 5000000.), a particular level of accuracy of the network is achieved.

**Step 5:** The purpose of the test is determining the system’s performance. The proposed model MLNN steps can be presented in the following algorithm.
Algorithm: Multilayer Neural Network (MLNN)

**Input:** 
- NoS // Number of samples with each class, 
- NoF // Number of features, 
- NoC // Number of classes, 
- FeatV() // Feature values for NoS within NoC, NoIn, NoHid, NoOut // No. of nodes in Input, Hidden, and Output layers, respectively, 
- MaxEpochs // Max number of epochs for training, 
- ErrorTh // Threshold value for error that when reached the training is stopped. 
- TS() // Total number of samples within each class to train with, 
- Weights() // Neural network weights resulted from training process.

**Output:** 
- Final Weights() // Final Neural Network weights after training with a size equal to (0 to NoIn * NoHid * NoOut), 
- Success_Count[] // Success count table that indicates how the tested sample will be distributed in each class.

**Begin**

**Step 1: Initialization**
Set xValues=0, yValues=0, tValues={[0,0] [1,1]} // target values., 
Max Epochs =1000, 
Max Validation checks = 6, Max Gradient = 1.00e-06 
bnn = new BackpropNeuralNet (NoIn, NoHid, NoOut). 
Set Nweights = NoIn * NoHid * NoOut, Max= 0, Epoch=0, TotalSample=0

**Step 2: Initialize weights**
For i = 0 to Nweights
initWeight(i) = 0.0.0.1 * Rnd() // initial weights with random values where Rnd() is a function return random value between (0,1)
End For
Call bnn.SetWeight(initWeight)

**Step 3: Find Maximum Training Samples Number**
For i = 0 to NoC
If ( TS[i] > Max ) then Max= TS[i]
End If
End For

**Step 4: Training**
While (Epoch < MaxEpochs) Do
For Sample = 0 to Max 
For Class = 0 to NoC 
If ( Sample < TS[class] ) then
Increment Total Sample by 1
End If
End For
For Feature = 0 to NoF 
Xvalue[Feature] = FeatV[class, Sample, Feature]
End For
yValues = call bnn.ComputOutput(xValues)
Error = Error + call bnn.GetError(tValues[class],yValue)
Call bnn.UpdateWeights(tValues[class])
End If
End For
End For
Error = Error/TotalSamples
If ( Error ≥ ErrorTh ) then
FinalWeight = call bnn.GetWeights()
END If
ErrorTh[Epoch]= Error
If (Error ≥ ErrorTh) then 
Increment Epoch by 1
END If
End While // Training End

**Step 5: Testing**
For Class = 0 to NoC 
For Sample = 0 to NoS[Class] 
For Feature = 0 to NoF 
Xvalue[Feature] = Feature[ Class,Sample,Feature]
End For
yValues= Call.bnn.ComputeOutput[xValues]
For i = 0 to values.Length
yValue[i] = Round[yValue[i]]

3751
3.5 Performance Evaluation of the Classification Model

To measure the classification performance progression in this research, three measures were used, namely accuracy, sensitivity, and specificity, as they are common procedures in the medical field. These dimensions were obtained from the confusion matrix which was a fundamental source to evaluate the binary classification models. The estimation measures utilized in the presented study are explored as follows.

A. Accuracy

An accuracy of a measurement system is represented by the number of adequately-diagnosed patients in the dataset, i.e. if it is negative or positive. This might be evaluated with the mathematical formula in Equation (4) [18].

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \ldots (4)
\]

Where TN represents True Negative and FP represents False Positive.

B. Sensitivity

Sensitivity is the ability of adequately choosing the positive patients subjected to the disease; i.e. the probability to have the disease (positive test) [18]. Sensitivity was utilized for determining any type of diseases in the classification approach; thus, the test sensitivity is equal to the proportion of the number of TP patients to the total number of sick individuals in population. It might be mathematically represented via Equation (5).

\[
Sensitivity = \frac{TP}{TP + FN} \quad \ldots \ldots \ldots \ldots (5)
\]

C. Specificity

It is the measurement of the percentage of adequately-predicted negative samples to the actual total number of negative samples. It is a significant measurement in the medical field, since it might help correctly detecting the subjects who are not carrying the disease. It can be calculated by using the following Equation (6):

\[
Specificity = \frac{TN}{TN + FP} \quad \ldots \ldots \ldots (6)
\]

A positive result in any one of the classification tests with high specificity range is sufficient to take decisions in any disease type [18].

4. Experimental Results

In this research, the proposed system was implemented using the programming MATLAB R2018b. The system specifications were as follows: The processor is Intel(R) Core (TM) i7 Cpu@1.73GHz. The installed memory is 8 GB. The system type is 64-bits operating system. The nnstart (Neural Network Start) toolbox is used in the current work. MLNN was implemented to classify diabetes disease with the use of IDPD. The results of utilizing MLNN for distinguishing between non diabetes and diabetes indicated extremely excellent network capabilities. The model has the ability of classifying 98.6% of cases in the testing set. Figure (3) shows the NN Training State. In addition, the value of best validation performance was 0.05719 at epoch 23, as can be seen in Figure (4). The Cross Entropy (CE) is the difference between targets and outputs. Low values are better, whereas zero indicates no error.

The data was categorized into 15% testing set, 15% validation set, and 70% training set. In the input layer, the neurons receive the data and transfer them to neurons in the hidden layer that has 16 neurons, through the weighted links. The sigmoid function is used to train back
propagation and adjust the weights through its algorithm. Lastly, the accuracy of classification was measured. Table (6) provides the performance accuracy measurement. The weights were altered in each one of the rounds in a way that the error is expected to be lower than in the previous round. Figure (3) shows Neural Network Training State when the gradient is 0.015096 at epoch 29 and the number of validation checks is 6, also at epoch 29.

![Figure 3](image)

**Figure 3**-Neural network training state in one hidden layer using IDP dataset.

Figure 4 shows the cross entropy versus the epoch number. The green line specifies the validation error, while the blue line represents the training error. In the target network, which has 16 neurons in the hidden layer, the minimum validation error occurred at epoch 23, as shown by the circle. The network parameters were saved at this point. Since the classification techniques had discrete target values, regression analysis was not useful for result validation.

![Figure 4](image)

**Figure 4**-Neural network training performance in one hidden layer using IDP dataset.

The error histogram of Neural Network training with 20 bins (i.e. Errors= Target – Outputs) is shown in Figure (5), where the minimum error was obtained and the network has stopped.
Figure 5: Histogram of neural network training error in one hidden layer using IDP dataset.

Figure 6 shows the operating characteristics of the Neural Network training receiver which includes true positive rate and false positive rate for training, testing, and validation.

Figure 6: Operating characteristics of neural network training receiver in one hidden layer using IDP dataset.
According to Table (4), the cross entropy of training is 0.1916, while the error is 1.0%. In addition, the cross entropy of validation is 0.4370, while the error is 2.4%, and the cross entropy of testing is 0.4566, while the error is 2.4%.

**Table 4** - The cross entropy (CE) and error values for the training, validation, and testing in one hidden layer using IDP dataset

<table>
<thead>
<tr>
<th>Results</th>
<th>Samples</th>
<th>Cross Entropy</th>
<th>%Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>198</td>
<td>0.1916</td>
<td>1.0%</td>
</tr>
<tr>
<td>Validation</td>
<td>42</td>
<td>0.4370</td>
<td>2.4%</td>
</tr>
<tr>
<td>Testing</td>
<td>42</td>
<td>0.4566</td>
<td>2.4%</td>
</tr>
</tbody>
</table>

The performance measurement elements of the proposed model of MLNN consist of five metrics, including epoch, time, performance, gradient, and validation checks. The performance measurements for the MLNN are shown in Table (5).

**Table 5** - The performance measurement elements of the FFNN model in one hidden layer using IDP dataset

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch [ 0 1000]</td>
<td></td>
<td>29 Iterations</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td>0: 00:01</td>
<td></td>
</tr>
<tr>
<td>Performance [0.618 0.00]</td>
<td></td>
<td>0.0593</td>
<td></td>
</tr>
<tr>
<td>Gradient [1.29 1.00e-06]</td>
<td></td>
<td>0.0151</td>
<td></td>
</tr>
<tr>
<td>Validation checks [ 0 6]</td>
<td></td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

The final results of diabetes Type II classification were obtained by using all confusion matrix measurements and demonstrated that the proposed model reached the highest accuracy of 98.6%, sensitivity of 95.2%, and specificity of 99.2% with the three features, as depicted in Table (6).

**Table 6** - The performance accuracy measurement in one hidden layer using IDP dataset

<table>
<thead>
<tr>
<th>Performance Evaluation</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>True negative (TN)</td>
<td>238</td>
</tr>
<tr>
<td>True positive (TP)</td>
<td>40</td>
</tr>
<tr>
<td>False negative (FN)</td>
<td>2</td>
</tr>
<tr>
<td>False positive (FP)</td>
<td>2</td>
</tr>
<tr>
<td>Accuracy (TP+TN)/(TP+TN+FP+FN)</td>
<td>98.6%</td>
</tr>
<tr>
<td>Specificity (TN/FP + TN)</td>
<td>99.2%</td>
</tr>
<tr>
<td>Sensitivity (TP/TP + FN)</td>
<td>95.2%</td>
</tr>
</tbody>
</table>
Figure 7 shows the results of the confusion matrix, which illustrate that the network achieved 98.6% accuracy. Thus, 40 samples were correctly classified as non-diabetics, i.e. a value of 0, corresponding to 14.2% of all 282 samples. While, 238 samples were correctly classified as diabetic, i.e. a value of 1, corresponding to 84.4% of all samples. The number of the non-diabetics samples that are incorrectly classified as diabetic was 2, (0.7%). While, 2 diabetics were incorrectly classified as non-diabetic (0.7%).

![Confusion Matrix]

**Figure 7-All confusion matrixes for MLNN.**

Out of 42 non-diabetics predictions, 95.2% are correct and 4.8% are incorrect. Also, among all the 240 diabetics’ predictions, 99.2% are correct and 0.8% are incorrect.

For the performance improvement of our proposed model MLNN, different ratios of training, validation, and testing sets were applied and showed different results, as illustrated in Table (12).

**Table 12-Performance results for MLNN based on different ratios of training, validation, and testing sets in one hidden layer using IDP dataset**

<table>
<thead>
<tr>
<th>No. of Samples of IDPD</th>
<th>Ratio of Training</th>
<th>Ratio of Validation</th>
<th>Ratio of Testing</th>
<th>Accuracy%</th>
<th>Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>282</td>
<td>50%</td>
<td>25%</td>
<td>25%</td>
<td>98.9%</td>
<td>1.1%</td>
</tr>
<tr>
<td>282</td>
<td>60%</td>
<td>20%</td>
<td>20%</td>
<td>97.5%</td>
<td>2.5%</td>
</tr>
<tr>
<td>282</td>
<td>70%</td>
<td>15%</td>
<td>15%</td>
<td>98.6%</td>
<td>1.4%</td>
</tr>
<tr>
<td>282</td>
<td>80%</td>
<td>10%</td>
<td>10%</td>
<td>98.7%</td>
<td>1.8%</td>
</tr>
<tr>
<td>282</td>
<td>90%</td>
<td>5%</td>
<td>5%</td>
<td>98.9%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>

The main contribution for our proposed model is that it is considered as the first work implemented on Iraqi diabetes patients dataset. Therefore, a comparison between the results obtained by using the proposed model could not be performed with other studies, which were applied only on the public PIMA Indian dataset.
5. Conclusions
An automatically diabetes diagnosing model was proposed, which uses multilayer neural network classification technique. To detect Type II diabetes, three features were selected for the network as an input and one variable class as an output. The model was applied on 282 samples of the Iraqi Diabetes Patients Dataset. For evaluating the classification performance, three measures, namely accuracy, sensitivity, and specificity, were used. The experimental results obtained by applying MLNN methodology for distinguishing between non-diabetics and diabetics showed extremely excellent network capabilities. The model was able to classify 98.6% of the cases. For future work, we are planning to increase the number of the features that are related to the Iraqi Diabetes Patients Dataset. In addition, a combination of machine learning algorithms can be used to build an improved diabetes diagnosis model.

References
