

**EAST WEST UNIVERSITY**

**Project report**

**Course Profile  
Course code: 365**

**Course Title: Artificial Intelligence**

**Submitted To**

**Department of CSE**

**Ease West University  
 Jesan Ahammed ovi**

**Submitted By**

|  |  |
| --- | --- |
| **Md. Tanvir Parvez** | **2017-2-60-088** |
| **Debobroto Ghosh** | **2017-2-60-125** |
| **Md. Atiqur Rahman** | **2017-2-60-087** |

**Problem statement:**

We have a dataset of some Diabetes patient regarding the issue of Diabetes. In our dataset we have 8 attributes information. With the help of that information, we are going to find out if a person is at the risk of Diabetes or not. Data accuracy depicts what is the chance of a person to have diabetes.

**System Requirements:**

Modern Operating System: x86 64-bit CPU (Intel / AMD architecture), 8 GB RAM, 5 GB free disk space.

**System design:**

Step 1: A set of inputs enter the network through the input layer and are multiplied by their weights.

Step 2: Each value is added to receive a summation of the weighted inputs. If the sum value exceeds the specified limit (usually 0), the output usually settles at 1. If the value falls short of the threshold (specified limit), the result will be -1.

Step 3: A single-layer perceptron uses the concepts of machine learning for classification. It is a crucial model of a feedforward neural network.

Step 4: The outputs of the neural network can then be compared with their predicted values with the help of the delta rule, thereby facilitating the network to optimize its weights through training to obtain output values with better accuracy. This process of training and learning generates a gradient descent.

Step 5: In multi-layered networks, updating weights are analogous and more specifically defined as backpropagation. Here, each hidden layer is modified to stay in tune with the output value generated by the final layer.

**Implementation:**

1. data = pd.read\_csv('diabetes.csv').fillna(0)

//We are reading the dataset and if there is a missing value in any column, it fill that column by inserting 0.

2. data['Outcome'].value\_counts()

//counting the Outcome value

3. x=data.drop('Outcome',axis=1)

//delete Outcome column from data and store in x

4. y=data['Outcome']

//copy Outcome column to y

5. trainX, testX, trainY, testY = train\_test\_split(x, y, test\_size = 0.2)

// we are splitting the dataset, one portion for training and the other portion for testing. We are only test a small group of data (20%), for this reason it multiplies the whole data set by a test size thus it can test a small group of data.

6. sc=StandardScaler()

7. scaler = sc.fit(trainX)

8. trainX\_scaled = scaler.transform(trainX)

9. testX\_scaled = scaler.transform(testX)

StandardScaler is that it will **transform your data such that its distribution will have a mean value 0 and standard deviation of 1**

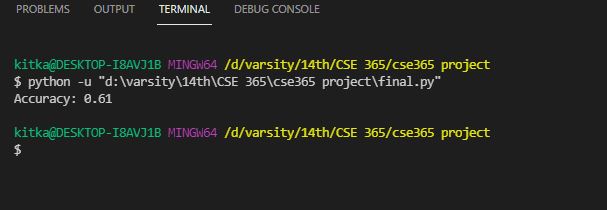
10. main\_fun = MLPClassifier(hidden\_layer\_sizes=(150,100,50),max\_iter = 1000,activation = 'logistic',solver = 'sgd')

// Here activation function is logistic or sigmoid function which results the data in the range of [0-1]. Here total iteration is 1000 and there are 3 hidden layers, and hidden layers has 150,100 and 50 as inputs. And the learning rate is 0.01.

11. main\_fun.fit(trainX\_scaled, trainY)

12. y\_pred = main\_fun.predict(testX\_scaled)

**Testing result:**

****

Here the accuracy of the data set is 0.61 which means if we consider randomly n number of data the output will show the probability of diabetes of a person after analyzing these data.

**Future Scope:**

There are some limitations in feed forward neural networks. It only works for data. It cannot identify the picture or any 2D array problem.

* The simplified architecture of Feed Forward Neural Network offers leverage in machine learning.
* A series of Feedforward networks can run independently with a slight intermediary to ensure moderation.
* The network requires several neurons to carry out complicated tasks.
* The handling and processing of non-linear data can be done easily with a neural network that is otherwise complex in perceptron and sigmoid neurons.
* The excruciating decision boundary problem is alleviated in neural networks.
* The excruciating decision boundary problem is alleviated in neural networks.