# Implementation of Backpropagation Neural Network in classification of Diabetes

Nithin Sai Jalukuru
School of Engineering and Applied Sciences
University at Buffalo, Buffalo, NY, USA
njalukur@buffalo.edu

**Abstract** : The classification of diabetes in the current work is carried out utilizing neural networks (with backpropagation) and a variety of input features. The prediction method for diabetes classification with outliers and missing values in data with class imbalance was the focus of this project. By using an adaptive synthetic sampling technique (ADASYN), the performance of the prediction model was less negatively impacted by class imbalance. ANN classifiers (with backpropagation learning) are then used to create assessments and predictions. According to experimental findings, using a neural network model helped this experiment achieve a greater accuracy of 81%.

**Keywords** : Neural Network, Kernels, Binary classification, Adaptive Sampling, ANN, Backpropagation.

## 1.INTRODUCTION

Similar to other supervised learning algorithms, neural networks also learn to map an input to an output based on a training collection of instances of (input, output) pairings. Neural networks in particular carry out this mapping by processing the input through a number of transformations. A neural network is made up of layers, and each layer is made up of units (also known as neurons), as shown in Fig.1:

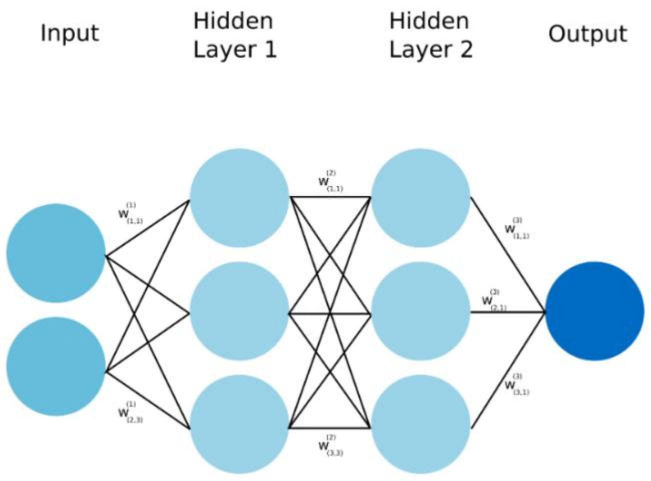


**Fig1**: Representation of Neural Network

### 1.1 Backpropagation

The fundamental idea behind backpropagation is to backpropagate errors from internal hidden layers to units of the output layer in order to adjust the weights and achieve *lower error rates*.

It is considered a practice of fine-tuning the weights of neural networks in each iteration. A trained neural network will be more robust and generalizable if the weights are tuned properly to provide the least possible loss.

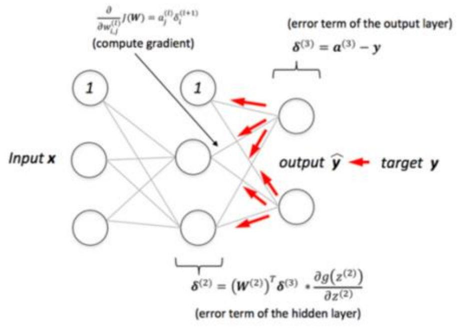

$\frac{\partial}{\partial w^{(l)}_{i,j}} J(W) = a^{(l)}_j \delta^{(l+1)}_i$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
Input x
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} \cdot \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

**Fig2**: Backpropagation representation (Source: Machine learning Greek)

### 1.1 Basic Steps in Backpropagation

Two passes through the computational graph:
→forward pass: compute the outputs of each layer.
→backward pass: compute the gradients of parameters in each layer, based on results from the forward pass

Backpropagation does this algorithm in an easy and inexpensive way.

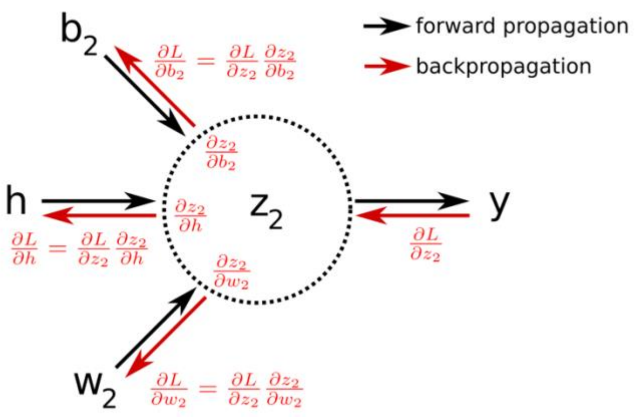

forward propagation
backpropagation
$b_2$
$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial b_2}$
$\frac{\partial z_2}{\partial b_2}$
h
$\frac{\partial z_2}{\partial h}$
$z_2$
y
$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial h}$
$\frac{\partial L}{\partial z_2}$
$\frac{\partial z_2}{\partial w_2}$
$w_2$
$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial w_2}$

**Fig3**: Working of Backpropagation (Source :GitHub/Romaintha repo)

### 1.2 Limitations of backpropagation Neural Network

*a)* The backpropagation technique calculates the derivatives of each neuron in the network for each backward iteration. The derivatives of the neurons that are dropped are calculated and then dropped even when dropout layers are applied.

*b)* The error function is anticipated to be convex. Backpropagation may become stuck in a local optimal solution for a non-convex function.

## 2. EXPERIMENTATION

### **1.1** Methodology

For the Backpropagation model implementation, a data pipeline structure is constructed as illustrated in the accompanying figure 4.

Data collection and loading into the environment are both parts of data extraction After loading, the data has been labeled and cleaned so that the ML model can analyze it effectively. . In addition, feature extraction a few features/attributes have been extracted, and the most pertinent ones have been chosen to train the model.

The correct ML model has been selected, trained, and verified during the model selection, training, and validation processes by maximizing the more suitable assessment criteria.

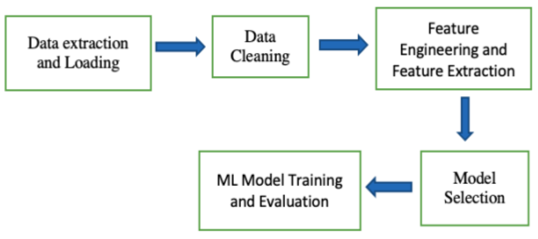

Data extraction and Loading
Data Cleaning
Feature Engineering and Feature Extraction
Model Selection
ML Model Training and Evaluation

**Fig4** : Data-pipeline

### **1.2** Data Extraction

The PIMA Indian Diabetes dataset, which was obtained from Kaggle, was used in the research study. Missing value and class distribution are typical unbalanced issues in medical data. The PIDD dataset consists of 768 instances with 8 attributes, only 113 data without missing values. Both samples with and without diabetes are included in this dataset.

There is a class imbalance in the dataset because there are 268 diabetes patients and 500 non-diabetic patients.

It incorporates 8 numerical features aspects, including the number of pregnancies, blood glucose levels, skin thickness, insulin levels, body mass index, diabetes pedigree function, and age (years), as shown in Fig. 5.

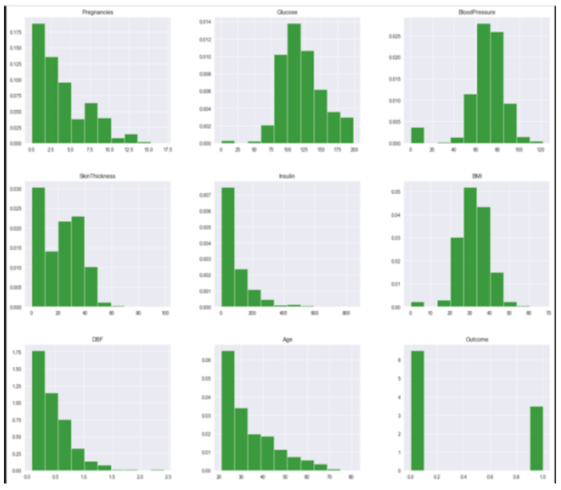


**Fig5**: Distribution of Attributes in PIDD

### **1.3** Data Cleaning and Feature Engineering

a) Proper attribute names have provided for easy interpretation and readability

b) Implemented IQR (Inter Quantile range) method to identify and removed the outliers. By doing this there will be no error in performance of ML model

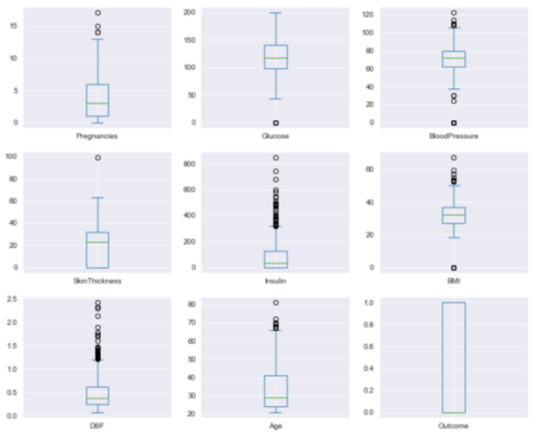


**Fig6**: Identification of Outliers

c) Implemented KNN imputer to handle the missing values in Glucose, SkinThickness, Insulin, BloodPrerssure and BMI features.

Found an Interesting visualization of interpreting the missing values in data

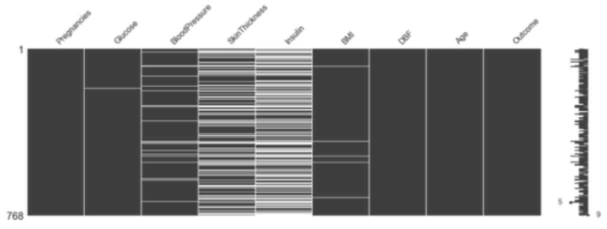


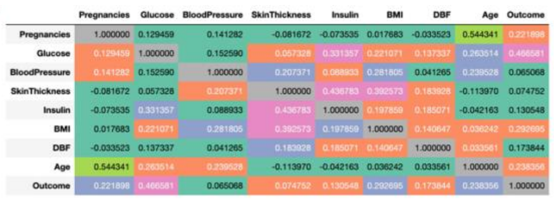
**Fig7**: Plot interpreting the Missing values in data

d) Performed feature standardization on the numerical data attributes ( Pregnancies, SkinThickness, Insulin, DBF, Age as they have right skewness , shown in fig 2) , where the feature values are re-scaled to have a mean of $\mu = 0$ and standard deviation $\sigma = 1$.

## 1.4 EDA (Exploratory Data Analysis) and Model selection

a) *Correlation between features*

The plot makes it clear that there is little dependence between the features (Low person Co-efficient) as shown in figure8



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DBF | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 | 0.263514 | 0.466581 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DBF | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238356 | 1.000000 |

**Fig8** : Correlation plot

b) *Frequency of Target Feature*

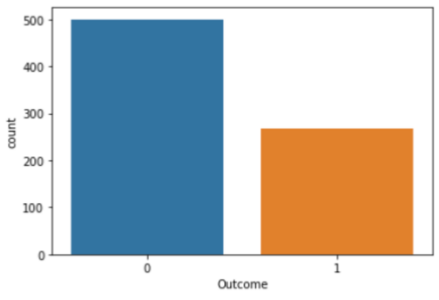


**Fig9**: Count Plot

There are 268 (outcome = 1) of diabetic patients and 500 (outcome = 0) of non-diabetic patients.

c) *Effect of age on diabetic's*

It is clear from fig. 10 that younger people are less likely to have diabetes than older ones.

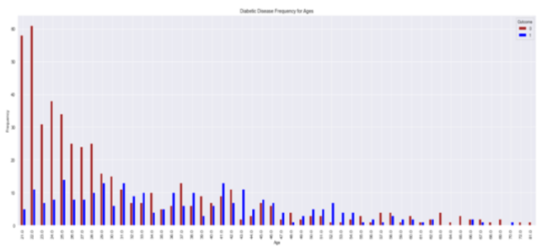


**Fig10** : count of each species

As there are only 268 samples of diabetics, An over sampling technique is applied. By employing the ADASYN approach, the original set of data points is contrasted with the data points from simple random oversampling, and the range of training samples is made bigger.

The main benefits of the synthetic data-generating algorithm ADASYN (Adaptive Synthetic) are that it avoids duplicating minority data and produces more data for examples that are "harder to learn."

Finally, the data is divided, with 80% going toward training and the remaining 20% going toward testing. Neural network machine learning model is ready with the data.

## 1.5 Backpropagation Modelling

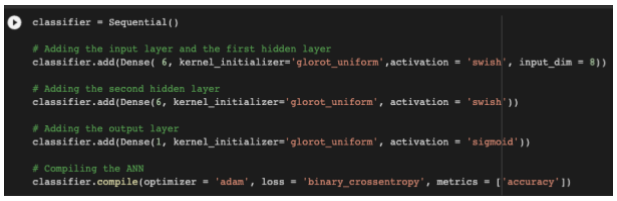
Backpropagation is a learning algorithm that artificial neural networks (ANN) employ to compute a gradient descent with respect to weights. The weights are updated backwards, from output to input, giving the algorithm its name.

To create a neural network using a backpropagation model using Keras. An ANN must first be initialized. The steps are as follows:

Added an input layer with an input dimension of 8 and no activation function, as well as a hidden layer 1 with a "*swish*" activation function and an output dimension of 6.

In the next step , a hidden layer 2 (output dimension is 6) has been added with "swish" activation function ( Swish *performs* better than ReLU on every batch size, indicating that the two activation functions' performance gap continues even when batch sizes are varied.)

In Final step , a output layer has been added with sigmoid as activation function and complied the ANN as shown



```
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense( 6, kernel_initializer='glorot_uniform',activation = 'swish', input_dim = 8))

# Adding the second hidden layer
classifier.add(Dense(6, kernel_initializer='glorot_uniform', activation = 'swish'))

# Adding the output layer
classifier.add(Dense(1, kernel_initializer='glorot_uniform', activation = 'sigmoid'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

**Fig11**: construction of Neural Network

ANN has been fit on training data with batch size 10 and epochs (This is the number of times the dataset will pass through the network, each time updating the weights. As the number of epochs increases, the network becomes better and better at predicting the targets in the training set) set to 100 and got a training accuracy of 0.7700 at Epoch 53/100.

Finally, this classifier is ready for predicting testing data and see the testing results in following section

## 1.6 Model Evaluation Metrics

Overall accuracy, recall, precision, f1 score, and AUC of the ROC curve are the most common evaluation measures for classification models.

False positives (FP) and false negatives (FN) are outcomes that were mistakenly classified by the model, while true positives (TP) and true negatives (TN) are outcomes of the positive class and negative class, respectively.

*a)* Overall Accuracy (OA): This is defined by the following the equation

$$\mathrm{OA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

The model could attain almost perfect overall accuracy if it consistently predicts the majority of classes.

*b)* The issue is more severe the more unbalanced the data. So, we require additional measurements. include Recall. It measures the proportion of accurately predicted positive classes to all positively categorized items.

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Recall is important when we believe False Negatives are more important than False Positives

*c)* Precision: It is the ratio of correctly predicted positive classes to all items predicted to be positive

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

It tells us how correct or precise that our model's positive predictions are. When we think False Positives are more significant than False Negatives, precision is crucial.

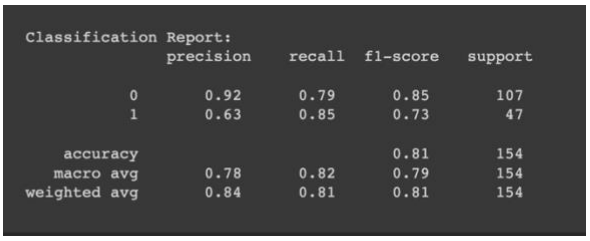
*d)* F1- Score : The F1-score is a single performance statistic that considers both recall and precision. It is also frequently referred to as the F-Measure. It is calculated by averaging the two metrics harmonically.

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

with values closer to one indicating better performance, and values closer to zero indicating poorer performance

## **1.7** Test Results

Let's look at the classification report



```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.79      0.85       107
           1       0.63      0.85      0.73        47

    accuracy                           0.81       154
   macro avg       0.78      0.82      0.79       154
weighted avg       0.84      0.81      0.81       154
```

**Fig12** : Classification report

Here, Recall is more important than the precision because of recall measures how many of the actual instances we were able to correctly predict. Because if a person is diabetic and model predict that the person has no diabetic then that will cause major mishap.

Also, there is a good tradeoff between recall and precision.

Also look at Confusion Matrix

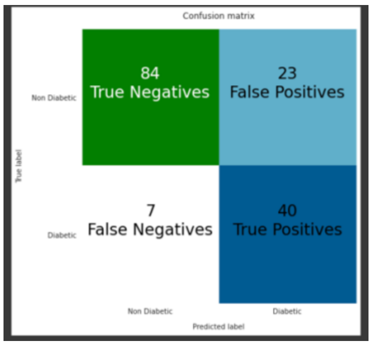


**Fig13** : Confusion Matrix

Backpropagation ANN has predicted as Non-diabetic for 7 people where they are actually diabetic, Which is a major concern.

Overall accuracy of the model is nearly 81% .

## 3. CONCLUSION

On the PIMA diabetes dataset, neural networks with backpropagation were used. The model had an accuracy of 81% and was very good at classifying the data into diabetics and non-diabetics. This score can be raised by experimenting with various activation methods, hidden layer counts, and epochs.

## 4. REFERENCES

1. .CSE 474/574 lecture slides (Prof. Chen)
2. Kaggle - Heart Disease classification
3. Shivani Yadav at. El. "A Neural Network based Diabetes Prediction on Imbalanced Data"
4. Swish Activation Function : https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820
5. https://github.com/romaintha/backpropagation/blob/master/Backpropagation.ipynb